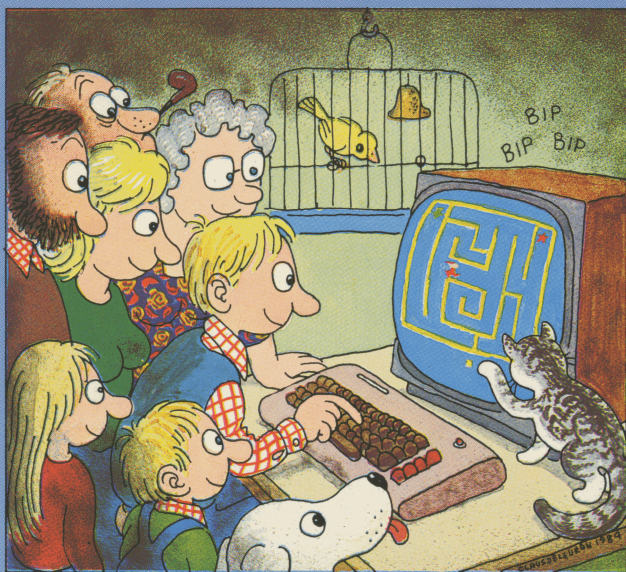


Erwin Neutzsky-Wulff

BASIC med COMMODORE 64



BORGEN

BASIC MED
COMMODORE 64

Af samme forfatter:

Dialog, roman 1971
Udstillingsbilleder, digte 1972
Tidsmaskinen, essay 1972
Seks hundrede seksogtres, digte 1972
Adam Harts opdagelser, roman 1972
Kærlighed, digte 1972
Etik, essay 1973
Parentes begynd, digte 1973
Anno Domini, roman 1975
Gud, roman 1976
Victor Janis og søn, roman 1977
Adam Hart og sjælemaskinen, roman 1977
Oiufael, roman 1977
Den treogtredvte marts, roman 1977
Havet, roman 1978
Indsigtens sted, roman 1980
Menneske, roman 1982
Mikrodatamaten. Programmering og Anvendelse,
en bog om ZX 81 BASIC, 1982
Programmering med Commodore BASIC, 1983
BASIC-programmering med Texas 99/4A, 1983

Erwin Neutzsky-Wulff

BASIC MED
COMMODORE 64

BORGEN

Copyright © 1984 Erwin Neutsky-Wulff

Alle rettigheder, herunder retten til erhvervsmæssig udnyttelse af bogens programeksempler, forbeholdes. Ingen del af denne bog må gengives, mangfoldiggøres ved fotokopiering eller på anden måde lagres i databank, forvandles eller transmitteres med brug af elektroniske eller mekaniske midler uden skriftlig tilladelse fra forlaget.

Omslag: Claus Deleuran

Illustrationer: Jesper Deleuran

Trykt hos Narayana Press

ISBN 87-418-7058-1

3. oplag 1985

Henvendelse til forfatteren ang. bogens indhold kan ske telefonisk hver torsdag kl. 12-13 og 17-19 på (01) 22 19 02.

Skriftlig henvendelse, samt henvendelse uden for ovennævnte tidspunkter frabedes venligst.

Forlaget vil gerne takke Tine Spang Nielsen og Bente Arildslund hos Commodore Data A/S i København for beredvillig hjælpsomhed i forbindelse med udlån af udstyr under bogens tilblivelse.

Til min elskede Elsebeth.

INDHOLD

FORORD 7

FØRSTE DEL:

VARIABLER OG PROGRAMMER

1. Variabler 13
2. Tastaturet 19
3. Programmer 24
4. Farveprogrammer 29
5. Goto 34
6. Input 39
7. If 44
8. Funktioner 48
9. Relationer 52
10. Logiske operationer 57

ANDEN DEL:

DATA OG SÆT

11. For-next-løkker 65
12. Strengdeling 69
13. Gosub 73
14. Sæt 77
15. Data 81
16. Strengsæt 85
17. Definerbare funktioner 88
18. Andre funktioner 92
19. Save 97
20. Spil 101

TREDJE DEL:

SKÆRM OG KARAKTERER

- 21. Bits og bytes 109
- 22. Systemvariabler 115
- 23. Skærmen 119
- 24. Bevægelse 123
- 25. Betingede udtryk 128
- 26. Skærmkoder 133
- 27. Bit-logik 139
- 28. Karakterer 143
- 29. Definerbare karakterer 148
- 30. Nyt karaktersæt 153

FJERDE DEL:

SPRITES, LYD M.M.

- 31. Højopløselig grafik 161
- 32. Grafisk afbildning 164
- 33. Sprites 168
- 34. Sprites i bevægelse 172
- 35. Sprites i spil 177
- 36. Tonen 182
- 37. Musik 185
- 38. Joystick 188
- 39. Diskette 191
- 40. Tidsmaskinen 196

APPENDIKS 205

INDEKS 219

FORORD

Meget er sket, siden jeg for fire år siden købte en ZX80 og hjemsøgte bibliotekerne for at finde baggrundsstof til en roman om en datamat, der vil være menneske. Det er selvfølgelig noget af en omvæltning for en skønlitterær forfatter med en menighed på et par tusind mennesker pludselig at blive solgt i femcifrede oplag, som tilfældet er med mine BASIC-bøger. Men det er i grunden ikke det mest mærkværdige.

Langt vigtigere har det været for mig at opdage den kulturrevolution, den personlige datamat repræsenterer, i hvor høj grad den betyder et opgør med hele vores magelige, næsten antiintellektuelle holdning til verden. *Her* er pludselig noget, vi synes er vigtigt og nødvendigt at lære, fordi vi forstår, at manglende interesse for disse ting i dag er livstruende analfabetisme i morgen.

Problemformulering er nøglen til programmering, men det er også nøglen til livet. Derfor har jeg, en god gammeldags romantisk blood-and-guts-poet, på dette område fundet en ganske betydelig udfordring. Så meget om filosofi.

Dette er en bog om COMMODORE 64, markedets uden sammenligning mest solgte maskine. Dens formål er at føre ham, der lige har anskaffet vidunderet og ladet den medfølgende brugsanvisning, som hverken kan lære ham BASIC eller noget andet sprog – hvilket selvfølgelig heller ikke er en brugsanvisnings opgave – gå i papirkurven. Han vil gerne vide, hvilken knap han skal trykke på først, og det får han at vide.

Herfra bevæger vi os via 40 lektioner frem til maskinens mest uigennemskuelige funktioner som sprites og lyd. De

40 afsnit falder i fire dele. Første del forklarer, hvad programmering overhovedet er for noget, og introducerer grundlæggende begreber som variabler, programmer, funktioner, etc. Vi lærer at arbejde med farver og grafik fra tastaturet, at fremstille og redigere enkle programmer med de grundlæggende ordrer og at forstå relationer og logik.

Anden del tager sig af programmering i lidt større stil med egentlig databehandling, sæt m.m. Vi ser på tekstbehandling, databaser og flere funktioner, samt finder ud af, hvordan vi gemmer det hele på kassettebånd.

I den tredje del beskæftiger vi os med skærmen og alle maskinens muligheder for (ikke mindst bevægelig) grafik. Det er begyndelsesgrundene for den, der vil lave spil med rumskibe og uhyrer, men også for en forståelse af sammenhængen mellem tastatur og skærm. Det vil sige, at vi skal gennem totalssystemet, PEEK og POKE, systemvariabler, betingede udtryk, skærmkoder, bit-logik og karaktergeneratoren. Vi vil opdage, at vi ikke blot kan jonglere med maskinens egne karakterer (forstørre dem op, f.eks.), men også definere vores egne. F.eks. »laver« vi det græske alfabet i otte linjer. Her som i resten af bogen dominerer selvfølgelig små grønne mænd og bombefly, hvis pædagogiske værdi næppe kan overvurderes. At læseren vil bruge bogen som spillemagasin uden at forstå den, er jeg egentlig ikke så bange for. Også det vil han nemlig uundgåelig lære noget af.

I fjerde og sidste del tager vi så fat på det tunge stof, der med resten af bogen som forudsætning nok ikke bliver så tungt endda. Vi skal se på højopløselig grafik med grafer i et koordinatsystem, og vi skal selvfølgelig lære at bruge de enestående sprites. Vi starter med et program, der lader os tegne den sprite, vi vil have, hvorefter maskinen selv regner sig frem til de nødvendige værdier (at gøre det med kvadret papir frarådes den, der ikke har hele døgnet til rådighed og vil beholde sin forstand lidt endnu). Derefter udforsker vi alle sprites særlige egenskaber, før vi kaster os ud i todæk-

keren for at udfordre den røde baron. På lydområdet går vi fra den enkelte tone til den flerstemmige melodi, og endelig gennemgås joystick og diskettestation for dem, der har sådan en.

Vigtigere er det imidlertid, at læseren på dette punkt skulle have baggrund for at iværksætte sin egen videre udforskning af maskine og sprog, som givetvis er ganske ubegrænset. Et lidt anderledes spil på en hel del K (se lektion 21) afslutter bogen sammen med nødvendige lister og tabeller.

Jeg har bestræbt mig på at forklare principperne på en sådan måde, at de er umiddelbart forståelige og giver læseren lyst til at bruge dem selv. Programmerne er til at lege med, men også til at lære af, forbedre og kassere, så læseren selv kan lave dem, som de skal være, og som han vil have dem. Hvis bogen kan leve op til dette krav, vil jeg være meget tilfreds.

Venlig hilsen og god fornøjelse
Erwin Neutsky-Wulff

FØRSTE DEL

Variabler og programmer

LEKTION 1

VARIABLER

**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM 38911 BASIC BYTES FREE

READY.

Sådan står der på skærmen, når vi tænder datamaten. Lad os starte med at se på, hvad det betyder.

BASIC er det sprog, maskinen taler, og som vi derfor må lære for at kunne tale med og bruge den. Det er det, der er denne bogs formål. Der findes også andre sprog for datamater, men BASIC er det uden sammenligning mest udbredte. De maskiner, der kun taler ét sprog, taler næsten altid BASIC. De, der taler flere, taler også BASIC. Det er en slags datamaternes engelsk, verdenssproget for computere.

COMMODORE 64 BASIC V2 er en dialekt af dette sprog. Når et firma fremstiller datamater i større stil, har det gerne sin egen dialekt af BASIC. Det hænger blandt andet sammen med den bestemte måde, dette firma bygger sine maskiner på. Jamen kan denne bog så lære os at tale med andre end firmaet Commodores maskiner? Ja, på samme måde som en sjællænder forstår en fynbo. Skifter man til en anden maskine, skal man lige vænne sig til den nye dialekt, præcis ligesom hvis en sjællænder flyttede til Fyn. Men sæt, han slet ikke kunne dansk?

Med skiltet COMMODORE 64 BASIC V2 angiver maskinen altså for os, hvilket sprog og hvilken dialekt den taler.

Hvad nu med 38911 BASIC BYTES FREE?

Hvis du har prøvet en billig lommeregner, ved du, at hvis vi skal regne ud, hvad $12 \cdot 34 + 56 \cdot 78$ er, må vi først lade den regne ud, hvad $12 \cdot 34$ er, og skrive resultatet ned, og så senere lade den lægge det til resultatet af den anden udregning. Den kan ikke huske det første resultat, mens den regner det andet ud. Det kan til gengæld lommeregnerne med hukommelse, og det kan en gigantlommeregner som COMMODORE 64 naturligvis også. Men hvor meget kan den så have »i hovedet« på en gang, hvad er dens hukommelseskapacitet? Den måler vi i BYTES, og hvad dette helt præcis betyder, skal vi vende tilbage til i en senere lektion. Her skal vi blot lidt løst konstatere, at en BYTE cirka svarer til et bogstav. Hvis du altså skal have maskinen til at huske en tekst på 100 bogstaver, skal du bruge godt og vel 100 bytes. 38911 BASIC BYTES FREE er maskinens måde at fortælle os, hvor meget hukommelse vi har til rådighed, som er til »fri« afbenyttelse for os, nemlig 38911 bytes. Denne hukommelse bliver så efterhånden »optaget«, når vi sætter datamaten til at huske forskellige ting.

Endelig står der READY, »parat«. Det vil sige, at maskinen er parat til at snakke med os og modtage ordrer. Den er ikke midt i en eller anden gigantisk udregning, som den helst ikke vil forstyrres i, den har faktisk ikke noget at lave. Lad os sætte den i arbejde.

Den kan altså huske. Udmærket, vi sætter den til at huske nogle tal for os. Den kan få lov til at huske vores telefonnummer: 123456. Hvordan gør vi nu det? Ja, vi kunne simpelt hen indtaste det, men hvad så, når vi skulle have fat i det igen, og det helst skulle være vores nummer og ikke et af de andre hundreder, vi havde tastet ind i maskinen? Ja, så skulle der gerne være navn på. Så vi skriver i stedet

PETER=123456

Lagde du mærke til den lille klods, der står og blinker? Det er *markøren*. Den kan altid vise os, hvor vi er på skærmen, det vil sige, det næste bogstav eller ciffer (den næste *karakter*), vi skriver, vil komme til syne der, hvor markøren nu er.

Hvad nu? Der synes ikke rigtig at ske noget. Nej, for maskinen kan jo ikke vide, om du er færdig. Måske kommer der flere cifre. Du må altså forklare maskinen, at der ikke kommer mere, at din ordre er færdig, og at den gerne må udføre den. Det gør du altid med tasten RETURN. Læg mærke til, at du ikke skal *skrive* RETURN. Det er en fast tast, der bare skal trykkes ned.

Nu kom skiltet READY igen frem på skærmen. Ordren er udført, maskinen husker dit telefonnummer, og den er parat til at modtage din næste ordre. Læg mærke til, at der stadig står 38911 BASIC BYTES FREE. Dette er selvfølgelig allerede en gammel oplysning, men den forsvinder ikke fra skærmen, så længe der er plads til, at den bliver stående. Det kan være meget praktisk at have tidligere resultater stående på den måde.

Hvordan får vi nu vores telefonnummer frem igen?

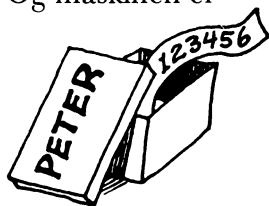
Jo, vi skriver

PRINT PETER

Det betyder skriv (egentlig tryk) på skærmen det, der står opført i maskinen som PETER, eller, om du vil, luk skuffen med PETER på op og skriv det tal, du finder i den. Huskede du RETURN? Så skulle tallet gerne vise sig. Og maskinen er igen READY.

Vi kunne godt have nøjedes med

PRINTPETER



Prøv det (husk RETURN). Et mellemrum optager også plads. I denne bog vil vi prøve at lære at undvære dem, også

fordi det er almindelig praksis, og du ellers vil få svært ved at forstå de ordrer, andre har skrevet.

Vi kunne også have nøjedes med

$P=123456$

I al fald indtil vi skal gemme nummeret på en person, hvis navn også begynder med P. Pouls nummer kan så blive oplagret som

$PO=789012$

eller

$P2=789012$

Når vi ikke har brug for at huske Peters telefonnummer mere, men i stedet skal huske, hvad vi har sat parkeringsuret på, kan vi skrive

$P=1230$

Det vil sige, egentlig skriver vi $P=123\emptyset$. Vi bruger $\emptyset$ som nul for at skelne det fra O som i Oda. Det var altså klokken halv et. Nu er skuffen P selvfølgelig blevet tømt, og der er lagt et andet tal ned i den. P kan stå for eller indeholde hvad som helst. Det er en *variabel*. Det tal, den *værdi*, vi anbringer i den, siger vi, vi *tilskriver variablen*. I ovenstående eksempel har vi altså tilskrevet variablen P værdien 1230.

Hvad nu, hvis vi skal huske telefonnummeret på to personer ved navn Peter? Prøv

$PETER1=123456$

Og (efter RETURN, selvfølgelig)

PETER2=789012

Skriv nu (igen efter RETURN)

PRINTPETER1

Av. Vi fik nummeret på PETER2 i stedet for. Hvorfor? Fordi maskinen kun læser de to første karakterer i variabelnavnet og følgelig ikke kan skelne mellem PETER1 og PETER2, hvorfor den opfattede PETER2 som en ny værdi af PETER1. Derimod havde P1 og P2 selvfølgelig været i orden. Dette er altså endnu en fordel ved at bruge forkortede variabelnavne, vi bliver lettere opmærksomme på, hvis vi bruger den samme variabel to gange. I virkeligheden ville man naturligvis aldrig oplagre telefonnumre i en datamat på denne upraktiske måde. Vi skal senere se, hvordan man gør.

Læg mærke til, at vi nu har mistet den øverste del af den oprindelige tekst. Der var ikke plads til den længere.

Måske er du kommet ubesværet gennem disse første eksempler. Men du har muligvis også haft problemer undervejs. Prøv

PRUNTPETER

Maskinen svarer med

?SYNTAX

ERROR

READY.

Syntaksen er BASIC-sprogets regler. Når du ikke følger dem, fortæller maskinen dig det:

Syntaksfejl! Og så er den for resten parat til, at du indtaster det rigtige . . .

Du kunne selvfølgelig også med det samme have rettet din fejl. Skriv

PRU

Nu kan du få U væk ved at bruge slettetasten INST DEL. Tryk på den, og U forsvinder. Skriv videre, til du har det rigtige PRINTPETER, og tast RETURN, og du får dit 789012. DEL betyder DELETE, slet. INST står for INSERT, en anden funktion på samme tast, som vi vender tilbage til.

Lad os til sidst indtaste nogle væddeløbsheste og deres odds. Vi har

TAROK=3

TROFAST=97

og

SPRINTER=5

Hvad nu? Syntaksfejl igen? Javist, der er et S, et E og et R foruden et = for meget i PRINT. Maskinen opfattede din variabel som en forkert stavet ordre. Se, det var jo ikke sket, hvis du havde brugt TA, TR og SP, vel? Flere variabler i næste lektion!

NB! Prøv

PRINT123456

LEKTION 2

TASTATURET

Skriv

PRINTA

Da variabelen A ikke er tilskrevet nogen værdi, skriver maskinen 0. Men hvorfor skrev den i grunden ikke bogstavet »A«?

For at skelne variabelnavne fra *streng*, det vil sige udtryk, der indeholder bogstaver, som »COMMODORE 64« og »DATAMAT« eller »DER ER ET YNDIGT LAND«, forlanger syntaksen, at streng skrives i gåseøjne. Skal vi have skrevet et »A« på skærmen, skal vi altså skrive

PRINT"A"

Gåseøjnene får du ligesom på en almindelig skrivemaskine med skiftenøgle. Hold SHIFT (lige til venstre for Z) nede og tast 2.

Streng kan naturligvis også gemmes, nemlig i *strengvariabler*, ligesom tal eller »numeriske« størrelser gemmes i *numeriske variabler*. Navne på strengvariabler kræver endvidere et dollartegn efter sig. Eksempelvis er altså

123 et tal

»COMMODORE 64« en streng

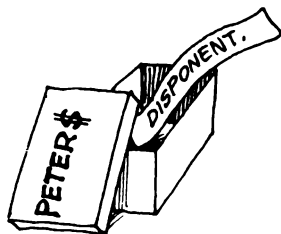
A en numerisk variabel

A\$ en strengvariabel

Skriv

PETER=41

og



PETER\$="DISPONENT"

Peters alder kunne opbevares i en numerisk variabel, fordi den var et tal, mens hans stilling, som var bogstaver, altså en streng, krævede en strengvariabel. Skriv nu

PRINTPETER\$PETER

Er du med?

Endelig kan et variabelnavn have procenttegn efter sig. Så er det en *heltalsvariabel*. Det er klart, at tallet 12.3456789 kræver mere plads end tallet 12. I variabelen (skuffen) A er der afsat plads til alle de mange decimaler. Hvis det så kun er 12, vi skal oplagre, spilder vi al den gode plads. Skriver vi nu i stedet for

A=12

ordren

A%=12

har vi dermed valgt en mindre rummelig og mere pladsbe-sparende skuffe. Den kræver simpelt hen mindre plads i hu-kommelsen. Til gengæld har vi så med denne formulering afskåret os fra at lade variabelen stå for et tal, der ikke er helt. Prøv

A%=12.3456789

og

```
PRINTA%
```

Variablen har »glemt« decimalerne. Prøv så

```
PRINT A
```

Ja, A er stadig 0. A% er en selvstændig variabel, der har lige så lidt med A at gøre som med A\$. Hvad er for resten A\$? Skriv

```
PRINT A$
```

Maskinen skriver ingenting, for variabelen er ikke tilskrevet nogen værdi. Strengen er »tom«. Skriv nu

```
CLR
```

og undersøg igen værdierne af PETER, PETER\$ og A%.

Strengvariablen er tømt, de numeriske nulstillede. Det er nemlig denne ordres funktion. CLR står for CLEAR, »rens« eller tøm alle variabler!

Ikke alene kan maskinen skrive bogstaver, den kan også skrive i flere farver. Du kan finde dem under tasterne 1-8:

- 1: BLK=BLACK=SORT
- 2: WHT=WHITE=HVID
- 3: RED=RØD
- 4: CYN=CYAN=LYSEBLÅ
- 5: PUR=PURPLE=VIOLET
- 6: GRN=GREEN=GRØN
- 7: BLU=BLUE=BLÅ
- 8: YEL=YELLOW=GUL

Hvis du prøver at taste dem, får du bare tallene. Du skal bruge CTRL (CONTROL = kontrol) og holde denne tast nede, mens du taster farven. Prøv at holde CTRL nede, mens du taster 1. Nu vil alt, hvad du skriver på skærmen, blive sort, til du igen skifter farve. Skriv på denne måde COMMODORE 64, dit navn, osv. Du kan nu fremhæve enkelte ord ved at skrive dem i en anden farve, men du kan også »vende dem om«, så de skrives f.eks. blå på sort i stedet for sort på blå. Dertil bruges tast RVS (REVERSE = omvend) ON (»slået til«) og RVS OFF (fra). Igen skal du bruge kontroltasten. Hold altså CTRL nede, mens du taster RVS ON.

Nu vil alt, hvad du skriver, blive »omvendt«. Mellemrummet, der normalt er »ingenting«, vil skrive en klods i den farve, du har valgt. Så længe du holder den lange tast nede, vil den trække en bjælke på skærmen i denne farve. Lige nu trækker den måske en sort bjælke. Slip nu tasten og hold CTRL nede, mens du taster 6 (GRN). Var det svært? Prøv at skrive COMMODORE 64. Det blev blå på grønt, ikke?

Men VIC kan mere end at skrive bogstaver. Prøv at holde SHIFT nede og taste S. Du får et blå hjerte på grøn baggrund. Lidt unaturligt, måske, men det kan vi nemt rette. Hold CTRL nede og tast RVS OFF. Nu fik vi »normal« skrift. Skift nu farve til rød (3 med CTRL nede). Ikke sandt?

Det, du fik, kan du finde under S-tasten på højre side. Tasten til venstre for SHIFT, der bærer Commodores bomærke (C=), holder du nede, når du skal have dem til venstre. Hold COMMODORE nede og tast B. To små røde kvadrater toner frem. »Commodore«-grafikken er måske knap så morsom som SHIFT-grafikken. Førstnævnte egner sig bedst til statistik, osv. Sidstnævnte kan vi bruge til at lege med. Prøv at trykke klør, spar, ruder og hjerter i de rigtige farver.

Hvor får vi nu de små bogstaver? Hertil kræves, at maskinen bringes i en tilstand, hvor den »læser« tastaturet på en anden måde. Hold SHIFT nede, mens du trykker på C=-ta-

sten. Læg mærke til, hvordan hele skærbilledet ændrede sig. I denne tilstand får du »normalt« små bogstaver og store med SHIFT, ligesom på en almindelig skrivemaskine.

Med C= holdt nede får du stadig »venstre« grafikken. Højregrafikken kan ikke nås i denne tilstand. Det er den mere »forretningsmæssige« side af tastaturet, vi her har fået frem. Vi skynder os at skifte tilbage med endnu et tryk på C= med SHIFT i bund.

Prøv farvetasterne med COMMODORE i stedet for CTRL!

Måske er det nu nærmest skærmen, der trænger til at blive »renset«. Hold SHIFT nede og tryk på CLR HOME-tasten. CLR står for CLEAR SCREEN, slet skærmen. Hvis du blot trykker på tasten uden SHIFT, er det HOME-funktionen, du får fat i. Markøren vender »hjem«, det vil sige til øverste venstre hjørne af skærmen. Det gør den selvfølgelig også med CLR, men den fjerner altså samtidig skærbilledet. Pas på ikke at forveksle CLR-tasten, der sletter *skærmen*, med *ordren* CLR, der sletter *variablerne*!

Du kan også flytte markøren andre steder hen ved hjælp af de to markørtaster mærket CRSR (CURSOR-markør). Pilene viser, hvad vej de flytter markøren, men skal du *op* og *til venstre*, skal du samtidig holde SHIFT nede. Trykker du en enkelt gang, flytter markøren et enkelt felt. Holder du tasten nede, kan du få den til at fare af sted, til du slipper.

På denne måde kan du rette overalt på skærmen. Skriv skærmen fuld. Bestem dig så for at rette et bogstav og lad markøren dække det. Nu behøver du kun at taste det bogstav, der skal erstatte det! Ret lidt rundt på skærmen på denne måde.

Eksperimentér med alt, hvad du har lært om tastaturet! I næste lektion skal vi nemlig i gang med at skrive et rigtigt program . . .

NB! Prøv

LEKTION 3

PROGRAMMER

Skriv ordrerne

```
PETER=41
PETER$="DISPONENT"
PRINTPETER$PETER
```

Maskinen skriver som før DISPONENT 41. Men sæt, vi nu ville have den til at gøre det samme igen? Så skulle vi skrive alle ordrerne en gang til. I stedet kan vi sætte ordrerne ind i et system af ordrer, en oversigt over, hvad maskinen skal gøre, bestående af ti eller hundrede ordrer. En sådan ordreliste kaldes et *program*. Vi kunne altså have

```
1 PETER=41
2 PETER$="DISPONENT"
3 PRINTPETER$PETER
```

Tallene foran hver ordre er *linjenumre*, der fortæller maskinen, i hvilken orden den skal udføre ordrerne, der nu er blevet *programlinjer* i programmet. Vi behøver altså ikke at taste linjerne ind i den rigtige orden, blot vi forsyner dem med de rigtige linjenumre. Og for at sikre os, at vi kan passe noget ind, vi har glemt, er det smart at lave et mellemrum på f.eks. 10 mellem de enkelte linjenumre. Vi får så

```
10 PETER=41
20 PETER$="DISPONENT"
```

30 PRINTPETER\$PETER

Prøv at skrive programmet op. Vi skal have RETURN efter hver linje. Mellemrummet mellem linjenummer og linjens indhold er ikke nødvendigt, men kan gøre programmet lettere læseligt.

Er du færdig? Der sker ikke noget. Nej, for et program må jo ikke »starte«, før det er skrevet færdigt. Det skal have en særlig ordre for at »køre«, og den hedder

RUN

Tast altså RUN fulgt af RETURN, og programmet går i gang. Tast RUN igen, og programmet kører endnu en gang. Du har givet maskinen en »fast« opskrift og dermed skrevet dit første program. Prøv nu

RUN 30

Nu skriver maskinen bare 0, for din ordre lød faktisk på, at maskinen skulle køre programmet *fra og med* linje 30, og så fik den ganske enkelt ikke værditilskrivningerne med. Men hvorfor var de der ikke fra sidste programkørsel? Fordi RUN *automatisk sletter alle variabler*. RUN betyder faktisk CLR & RUN! Hvad vil RUN 20 give? Prøv at regne det ud, og se så efter, om du havde ret. Kør programmet endnu et par gange. Nu kan du ikke længere se din PROGRAMLISTE. Skriv

LIST

og den kommer frem igen. Hvad vil LIST 30 give? Nemlig, linje 30 på skærmen. Prøv LIST 20-30. LIST 20-. LIST -20. Forvirret? Jo:

LIST giver hele programlisten

LIST N (hvor N er et eller andet linjenummer) giver linje N

LIST N- giver programlisten fra og med linje N

LIST -N giver programlisten til og med linje N

LIST M-N giver programlisten fra og med linje M til og med linje N

Det er selvfølgelig noget, du får mere brug for, når dine programmer bliver lidt længere! Skriv igen

LIST

Nu får du brug for, hvad du lærte om markørtasterne. Ret på denne måde 41 til 51. Men pas nu på! Dette retter kun tallet på *skærmen*. Du har ikke virkelig rettet i *programmet*, hvis du ikke følger rettelsen op med RETURN. Prøv selv!

Gå nu op og ret linjenummer 10 til 15. Tag så en LIST, men pas på, du ikke kommer til at skrive oven i noget andet, så du får LISTY eller noget andet for maskinen uforståeligt. Hov! nu er der både en linje 10 og en linje 15 med præcis det samme indhold. Javist, maskinen har jo en gang fået besked om linje 10, og det glemmer den ikke, medmindre den får besked om *det*. Du kan klare det med at skrive en »tom« linje, dvs. du taster simpelt hen linjenummeret på den linje, du vil af med, fulgt af RETURN. Slip af med 15 på denne måde og bed igen om listen. Nemt nok, ikke?

Du kan selvfølgelig også rette en linje ved at skrive den helt om. En ny linje erstatter altid en tidligere med samme linjenummer.

Men hvad nu, hvis du simpelt hen vil af med noget i en linje? Måske skal maskinen kun skrive værdien af PETER, og ikke af PETER\$? Ja, så skal vi af med PETER\$ i linje 30, og det går faktisk let, vi kan nemlig bruge vores slettetast fra før. Men først skal vi have markøren på plads. Sørg for at anbringe den oven på P i det andet PETER (som om vi lige

havde skrevet det, vi vil af med) og tast så DEL. Ikke blot forsvinder PETER\$ karakter for karakter, linjen trækker sig også sammen, så der ikke bliver noget »hul«. Læg mærke til, at også denne tast »fortsætter«, så længe du holder den nede, så pas på, den ikke løber fra dig! Nu hedder linje 30

30 PRINTPETER

Husk RETURN!

Vi kan også gå den anden vej og tilføje noget inde midt i en linje. Så er det, vi skal bruge den tidligere omtalte INSERT (indsæt), som vi får med SHIFT. Lad os f.eks. sige, at vi vil have linjen til at hedde

30 PRINT"A:"PETER

måske for at gøre opmærksom på, at tallet er en alder. Ja, så nytter det ikke noget bare at rette, for vi skal faktisk have linjen til at udvide sig og skaffe plads til yderligere 4 karakterer. Okay, vi rykker igen markøren hen, så den står direkte *efter* det, vi vil ændre på, altså oven i P i PETER. Hold nu SHIFT nede og tast INST 4 gange. Nu blev der plads, og du kan skrive

"A:"

fulgt af RETURN.

Nu vil du måske skrive et andet program, men så skal du først af med det, du har. Det har vi også en ordre til, nemlig

NEW

Prøv at sætte linjenumre på nogle af de tidligere eksempler og køre dem som programmer. Du kan også bruge f.eks.

CLR i et program. NEW i sidste linje i programmet vil slette det, så snart det er kørt. Eller prøv

```
10 PRINT"COMMODORE 64"  
20 RUN
```

Hvad gør maskinen? Jo, først skriver den »COMMODORE 64«. Så går den ned til linje 20, der giver den besked på at køre programmet forfra, det vil sige, den skriver igen »COMMODORE 64«, osv. Nu bliver problemet nærmest at standse programmet igen, men til det brug har vi tasten RUN STOP. Ved brug af den får vi beskeden

```
BREAK IN 10
```

dvs. »standset, mens jeg var i færd med at udføre linje 10«. På samme måde vil en syntaksfejl i et program give en besked, der gør opmærksom på, i hvilken linje fejlen findes.

Nu, da du ved, hvad et program er, kan vi gå over til nogle lidt mere farverige sådanne i lektion 4.

NB! Prøv

```
10 PRINT"OOOOOOOOOOOOOOOOOOOOOOOOOO  
OOOOOOOOOOOOOOOOOOOOOOOO"  
20 PRINT"*****  
*****"  
30 RUN
```

LEKTION 4

FARVEPROGRAMMER

Skriv

```
10 PRINT"64"
```

```
20 PRINT"FRA COMMODORE"
```

```
30 PRINT"DEN PERSONLIGE DATAMAT"
```

og kør. Hold CTRL nede og skriv 2. LIST programmet og kør det igen. Javel, men var det nu ikke mere praktisk, hvis f.eks. »64« og FRA kunne stå med grønt, »COMMODORE« med hvidt og *resten med sort*? Lad os prøve. Det kræver i al fald, at farvekontrollerne kommer inden for gåseøjnene, og for nu ikke at blande for meget sammen af det, vi lige har lært, kan vi blot skrive NEW og gå i gang igen. Skriv

```
10 PRINT"
```

Det er nu, vi skal have fat i farvekontrollen, som altså skal være den for grøn. Vi holder CTRL nede og taster 6. Nu fik vi en opadvendt blå pil på farvet baggrund. Tja, sådan ser farvekontrollerne altså ud inden for gåseøjne. Skriv linjen færdig og fremstil de andre linjer på tilsvarende måde. Linje 20 skal du altså skrive som

```
20 PRINT"FRA
```

mellemrum, CTRL nede, 2, og så resten af linjen. Farvekontrollen behøver altså ikke at ligge i begyndelsen af en linje.

Du fik et E for hvid og et (farvet) kvadrat for sort, ikke? Farvekontrollen vil vi for fremtiden i de programlister, der er aftrykt i denne bog, angive med et lille f foran tallet på den pågældende farvetast. Vores program ville altså komme til at hedde

```
10 PRINT"f664"  
20 PRINT"FRA f2COMMODORE"  
30 PRINT"f1DEN PERSONLIGE DATAMAT"
```

Du kan egentlig godt springe mellemrummene mellem linjenummer og linje over. Så snart du beder om programlisten, sætter maskinen dem selv.

Hvad med omvendt skrift? Efter præcis samme opskrift. Prøv at erstatte 20 med

```
20 PRINT"FRA f9COMMODORE"
```

eller

```
20 PRINT"FRA f2f9COMMODORE"
```

Vi kan også fremhæve »PERSONLIGE« i 30:

```
30 PRINT "f1DEN f9PERSONLIGE DATAMAT"
```

Men hov, nu blev »DATAMAT« jo også omvendt. Ja, for vi glemte at »slukke« vores omvender. Linjen skal altså hedde

```
30 PRINT"f1DEN f9PERSONLIGEf0 DATAMAT"
```

Altså:

En farvekontrol gælder i et program, til maskinen støder på en anden.

»Omvenderen« gælder linjen ud.

Mens vi er ved bogens programlister, skal grafikken jo også angives. Her vil et hjerte blive angivet som S, dvs. S SHIFT (skiftet S). Venstregrafikken, som vi får med COMMODORE-tasten, angives med et lille c. cB er altså det grafiske tegn til venstre på B, eller om man vil: B med cCOMMODORE holdt nede. Denne »kode« skulle afværge de fleste misforståelser. Prøv

```
10 PRINT"f2Sf4Qf1W"
20 RUN
```

Læg mærke til, at maskinen automatisk skifter linje efter hver PRINT-ordre. Tilføj nu et , til linje 10. Nu sprang den kun en kvart linje. Forsøg endelig med ; Let, ikke? Vil du have det til at gå langsommere, kan du holde CTRL nede. Selvfølgelig kan det gøres på en smartere måde. Det vender vi tilbage til. NEW og

```
10 PRINT"
```

Tryk nu på HOME (altså uskiftet), og et omvendt S kommer frem. Skriv

```
COMMODORE 64"
```

og

```
20 RUN
```

Kør programmet. Maskinen skrev COMMODORE 64 i øverste venstre hjørne, og dermed slut. Og dog. Programmet kører stadig og skal standses med RUN STOP. Det, der sker, er, at HOME, der fremstår i programmet som et om-

vendt S, bliver ved med at sende maskinen »hjem«, så den bliver ved med at skrive oven i det samme. Udskift nu 10 med

```
10 PRINT"
```

Tryk på CLR (altså skiftet), og et omvendt hjerte kommer frem. Skriv

```
COMMODORE 64"
```

og kør. Nu sletter maskinen også skærmen for hver gennemkørsel, og COMMODORE 64 blinker (meget hurtigt). Vi vil udtrykke HOME med lille h, og CLR med lille h med streg under. De to sidste eksempler ville vi skrive henholdsvis

```
10 PRINT"hCOMMODORE 64"  
20 RUN
```

og

```
10 PRINT"hCOMMODORE 64"  
20 RUN
```

Vi kan også bruge markørtasterne i et program. Skriv

```
10 PRINT"COMMODORE
```

Tast markør ned og markør til højre. Maskinen skriver omvendt Q og højreklamme. Skriv

```
COMMODORE"
```

I bogen vil vi skrive det som

```
10 PRINT"COMMODOREsøCOMMODORE"
```

s og ø står for »syd« og »øst«. Op og til venstre kommer så til at hedde n og v.

Du skulle nu være forholdsvis bekendt med tastaturet. I lektion 5 skulle vi så kunne tage lidt mere alvorligt fat på programmeringen af COMMODORE 64.

```
10 PRINT"f1QQQQQQQQQQQQQQQQQQQQQQQQQQ  
          QQQQQQQQQQQQQQQQQQQQQQQQ"  
20 PRINT"f9QQQQQQQQQQQQQQQQQQQQQQQQQQ  
          QQQQQQQQQQQQQQQQQQQQQQQQ"  
30 RUN
```

LEKTION 5

GOTO

Prøv

```
10 PRINTA  
20 A=A+1  
30 GOTO10
```

Hvad gør programmet? Ja, først skriver det værdien af A, som er 0, eftersom vi ikke har tilskrevet den nogen anden værdi. Så lægger den 1 til A. Dette er en lidt speciel form for værditilskrivning. Vi kunne kalde den »værdiforandring«. Linjen betyder: Tag tallet op af skuffe A, læg 1 til og læg det ned igen!

Linje 30 er også ny. GOTO omgår programmets naturlige nummerorden at tage programlinjerne i. I stedet for at gå videre til næste linje (og stoppe, eftersom der ikke er flere) giver linje 30 maskinen besked på at fortsætte kørslen i linje 10. Den skriver altså på ny As værdi (som nu er 1), lægger igen 1 til i linje 20, og så fremdeles. Programmet tæller. Hvorfor kunne vi nu ikke have brugt en linje 30, der hed

```
30 RUN
```

Også her ville vi være kommet tilbage til 10, som jo er programmets første linje. Godt nok, men vi ville være kommet dertil med slettede variabler. A ville igen være 0, og programmet ville skrive lutter nuller. Prøv at erstatte 10 med

```
10 PRINT"h"A
```

Vi kan også få maskinen til at skrive en tabel:

```
10 PRINTA
20 A=A+17
30 GOTO10
```

giver 17-tabellen. Maskinen lirer sin lektie af i en forfærdelig fart. Vi kan naturligvis altid stoppe den undervejs med RUN STOP eller blot sætte farten ned med CTRL. Eller vi kan lave et lidt smukkere program med

```
10 A=A+17
20 B=B+1
30 PRINT"f6"B"f1GANGE 17 ERf2"A
40 GOTO10
```

Eller lad den regne rentesregning. Vi indsætter 100 kroner til 9% rente og får

```
10 A=1984
20 B=100
30 PRINTA":B
40 A=A+1
50 B=B*1.09
60 GOTO30
```

Lad den bare køre lidt. I løbet af 30 sekunder vil den have udskrevet årligt indestående de næste 500 år!

Prøv at standse den ved år 2034. På 50 år er vores 100 kroner blevet til 7435.75. Det kan vist næppe betale sig. Men lad den så køre videre til 2084! Videre? Jeg mener vel forfra igen? Vist ikke. Tast

CONT

hvilket betyder »continue« = fortsæt, og programmet fortsætter. Nå, nu er den halve million da hjemme. Måske skulle vi lige snuppe 100 år til på den anden side, så vi kan blive hele millionærer. I 2184 har vi nemlig – hvad pokker betyder $3.05702913E+09$? Ingen panik! det betyder bare, at kommaet (som altså her er et punktum) skal flyttes ni pladser til højre. Vores indestående er altså kr. 3057029130 eller lidt over 3 milliarder. Mange bække små . . . Som det vil fremgå, kan maskinen regne, og vi kan da også bruge den til ganske almindelig købmandsregning. Prøv f.eks. ordren

PRINT123*456

Det var lidt lidt at skrive P-R-I-N-T for? Ja, måske, derfor kan du også nøjes med et spørgsmålstegn.

?123*456

vil virke præcis lige så godt. Regn selv med +, – (minus), * og / (divideret med). Husk blot, at maskinen *prioriterer*. Dermed menes, at maskinen fornuftigvis er ligeglad med, om du skriver

?123*456+789

eller

?789+123*456

Den kan aldrig finde på at lægge 789 sammen med 123 og gange resultatet med 456. *Den ganger og dividerer altid, før den lægger sammen og trækker fra.* Men hvad nu, hvis vi virkelig vil

have den til at lægge 789 og 123 sammen, og så gange med 456? Så klarer vi den med parenteser (ikke klammer):

$(789+123)*456$

Pilen på tasten til højre for * er til »potensopløftning«. I stedet for $7*7*7*7$ kan vi sige 7 opløftet til fjerde potens, det vil sige 7 ganget med sig selv fire gange. Vi taster 7 pil 4, og i bogen skriver vi

7^4

Maskinen opløfter altid til potens, før den indlader sig på nogen af de fire regningsarter. Prøv til sidst

```
10 A=1
20 PRINTA
30 STOP
40 A=A*2
50 GOTO20
```

Maskinen skriver 1 og standser så med beskeden

```
BREAK IN 30
```

Da den kom til STOP, kunne den ikke komme længere. Tast nu CONT, og programmet fortsætter med 2, hvor det igen stopper, og så fremdeles. Somme tider kan man undersøge et program for fejl ved at anbringe STOP-linjer i det. På den måde kan det køres »lidt ad gangen«. Erstat 30 med

```
30 END
```

Denne linje betyder, at programmet er slut (og er derfor overflødig i programmer, der kun skal ende i sidste linje) og

der kommer ingen meddelelse. Vi kan dog stadig få det til at fortsætte med CONT. Slet 30 helt og lad det køre. Ved 4.25352959E+37 standser det med beskeden

?OVERFLOW
ERROR IN 40

Længere kan maskinen nemlig ikke »tælle«. Så hvis du har mere end 42 billioner billioner billioner i banken, skal du købe en større datamat . . .!

NB! Prøv

```
10 A=1984
20 B=B+100
30 PRINTA": "B
40 A=A+1
50 B=B*1.09
60 GOTO20
```


LEKTION 6

INPUT

Prøv igen

```
10 A=A+17
20 B=B+1
30 PRINT"f6"B"f1GANGE 17 ERf2"A
40 GOTO10
```

Hvad nu, hvis vi vil have 18-tabellen i stedet for? Vi kan selvfølgelig rette i programmet, men det ville dog være ulige smartere, hvis det samme program kunne bruges til en hel række tabeller. Prøv

```
5 INPUTC
10 A=A+C
20 B=B+1
30 PRINT"f6"B"f1GANGE"C"ERf2"A
40 GOTO10
```

Det, der bliver lagt til i 10, er nu værdien af variablen C, og den kommer åbenbart fra linje 5. Men hvordan? Jo, INPUT-linjen betyder simpelt hen, at maskinen skal vente på, at et tal indtastes, og derefter tilskrive det som værdi til variablen C. Kør programmet. Der kommer et spørgsmålstegn frem på skærmen. Dette er maskinens måde at bede om en indtastning på. Indtast et tal, og programmet kører. Det blotte og bare spørgsmålstegn vil naturligvis forvirre den, der ikke

kender programmet. Derfor ville det måske også være en god idé at indføre en forklarende PRINT-linje, som f.eks.

```
4 PRINT"HVILKEN TABEL"
```

Men vi kan også inkludere den i selve INPUT-linjen og få

```
5 INPUT"HVILKEN TABEL";C
```

Prøv at køre det. Smukt, ikke? Her skulle maskinen altså »fodres« med et tal. Men det kunne også sagtens have været en streng:

```
10 INPUT"DU HEDDER";A$  
20 INPUT"DU FYLDER I 1984";A  
30 PRINT"DU ER FØDT"1984-A;A$
```

Læg mærke til semikolon mellem A og A\$. Uden det ville det være blevet til strengvariablen AA\$. Lad os prøve at slette det. Maskinen siger

```
?TYPE MISMATCH  
ERROR IN 30
```

Dette betyder blot, at vi har prøvet at trække en streng fra et tal, og det kan selvfølgelig ikke lade sig gøre. Ret programmet tilbage og kør igen, men svar nu omvendt på spørgsmålene, altså alder på spørgsmål om navn og omvendt. Maskinen havde ingen vanskeligheder med navnet (måske hedder du virkelig 91), men da du i stedet for det ventede tal indtastede bogstaver, protesterede den

```
?REDO FROM START
```

om igen soldat: Indtast nu bare et tal, og maskinen fortsæt-

ter. Vi kan strengt taget også bruge INPUT som en (noget omstændelig) måde at bremse løbske programmer på. Prøv

```
10 A=1
20 PRINTA
30 INPUTB
40 A=A*2
50 GOTO20
```

Hver gang maskinen når til INPUT-linjen, standser den med et spørgsmålstegn. Tast blot RETURN (der lader B beholde sin værdi, men det er jo lige meget), og programmet tager et bette nøk til!

Hov, hvordan sytten kommer du ud af det her igen? RUN STOP virker ikke. Jo, hvis du holder tasten nede og samtidig trykker på RESTORE! Samtidig bliver markøren blå (hvis den ikke var det i forvejen).

Læg mærke til, at maskinen sagtens kan tage mod flere INPUT i en linje:

```
10 INPUT"DATA";A,B$,C,D
20 PRINT A;B$;C;D
```

Kør programmet med RUN og indtast

```
12 (RETURN)
COMMODORE (RETURN)
34 (RETURN)
56 (RETURN)
```

Du kan spare på RETURNerne ved at indtaste det hele på én gang:

```
12,COMMODORE,34,56 (RETURN)
```

Prøv at indtaste

```
12,COMMODORE,34 (RETURN)
```

Spørgsmålstegn angiver, at der er endnu en variabel, der kræver påfyldning. Eller indtast

```
12,COMMODORE,34,56,78 (RETURN)
```

Maskinen oplyser:

```
?EXTRA IGNORED
```

dvs. »jeg ser bort fra det, der er tilovers«, som der ikke er nogen variabel til!

Vi kan nu få et næsten brugbart renteprogram ud af det:

```
10 INPUT"BELØB";A
20 INPUT"RENTEFOD I %";B
30 C=1984
40 PRINT"f1f9"C"f0f2"A
50 INPUT
60 A=A+A*B/100
70 C=C+1
80 GOTO40
```

Da INPUT-linjen i 50 kun er en pause, kan variabel helt undværes. Vi kan også sætte maskinen til at høre os i tabellerne:

```
10 INPUT"TABEL";A
20 B=B+1
30 PRINT"f1HVAD ER"B"GANGE"A
40 INPUT C
50 PRINT"SVARET ERf9"B*A
60 GOTO20
```

Selvfølgelig er det lidt irriterende at få at vide, hvad »svaret er«, når man lige selv har indtastet det. Bedre ville det være, hvis maskinen skrev BRAVO eller noget lignende, når vi havde svaret rigtigt. Det ville jo imidlertid kræve, at maskinen kunne se, om vi havde svaret rigtigt. Det må vi prøve i næste lektion.

NB! Prøv

```
10 INPUT A
20 B=B+A
30 PRINT B
40 GOTO 10
```

LEKTION 7

IF

I lektion 6 havde vi et program, der i princippet så sådan ud:

```
10 INPUT "TABEL";A
20 B=B+1
30 PRINT "HVAD ER "B"GANGE"A
40 INPUT C
50 PRINT "SVARET ER "B*A
60 GOTO 20
```

Det irriterede os, at maskinen ikke havde nogen mulighed for at opdage, om vi svarede rigtigt eller forkert. HVIS vi svarede rigtigt, SÅ skulle maskinen nemlig rose os. HVIS – SÅ hedder på engelsk – og i BASIC – IF – THEN. Vi kunne nu indsætte en linje 45, der hed

```
45 IFC=B*A THEN PRINT "BRAVO"
```



Hvis vi tilføjede

```
46 IFC=B*ATHENGOTO20
```

ville vi endda slippe for 50. Faktisk ville

```
46 IFC=B*ATHEN20
```

være nok, eftersom GOTO er underforstået umiddelbart efter THEN. Faktisk kan vi nøjes med en enkelt linje:

```
45 IFC=B*ATHENPRINT"BRAVO":GOTO20
```

Vi har her fået passet en ekstra linje ind i 45; oven i købet »rider« den på det samme IF – THEN som PRINT-linjen. Alt, hvad der kræves, er *et kolon mellem de to ordrer*, og de er én linje. Maskinen vil udføre alle de ordrer, der står efter THEN *i samme linje*. Eller hvad med

```
10 INPUT"TABEL";A
20 B=B+1
30 PRINT"HVAD ER"B"GANGE"A
40 INPUTC
50 IFC=B*ATHENPRINT"RIGTIGT":GOTO20
60 PRINT"NEJ –"B*A
70 GOTO20
```

Hvad sker der, hvis C ikke er =B*A? Ja, så går maskinen videre til 60 og skriver NEJ + det rigtige resultat.

Når betingelsen ikke er opfyldt, går maskinen videre til næste linje.

Dette program vil selvfølgelig fortsætte til dommedag. Det vil fortsætte med at udfritte os med hensyn til, hvad 97 gange 17 er, og hvad 2345 gange 17 er. Men det behøver det jo egentlig slet ikke. Tilføj

```
25 IFB=11THENEND
```

eller bedre

```
25 IFB=11THEN80
```

Nu bliver vi nemlig sendt til en endestation, der ligger uden for karrusellen (»løkken«), der så kan hedde

```
80 PRINTD"RIGTIGE"
```

Hvad hulen er D? Jo, det er vores »tæller«, der tæller, hvor mange gange vi svarer rigtigt. Derfor skal vi også have udvidet vores 50 til

```
50 IFC=B*ATHENPRINT"RIGTIGT":D=D+1:GOTO20
```

eller på menneske-dansk: Hvis der svares rigtigt, skal maskinen skrive RIGTIGT på skærmen, tælle en på tælleren (»en til rigtig!«) og gå videre til næste spørgsmål (B 1 højere).

Vi har nu fået et program med en meget lang linje og et par meget korte. Kunne vi ikke klistre nogle af de korte sammen til lange med kolon? Jo, sagtens. Og eftersom programmet er færdigt, og der ikke skal passes mere ind, kan vi også spare lidt på linjenumrene. Læg så farver på, og du har noget, der ligner et rigtigt program:

```
1 INPUT"TABEL";A
2 B=B+1:IFB=11THEN5
3 PRINT"f1HVAD ER"B"GANGE"A:INPUTC:IFC=
  B*ATHENPRINT"f2f9RIGTIGT":D=D+1:GOTO2
4 PRINT"f6f9NEJ -"B*A:GOTO2
5 PRINT"f5f9"D"RIGTIGE"
```


Hvorfor kan 60 og 70 ikke klistres bag på 50? Nej, for så bliver de jo også konsekvenser af $C=B*A$. Ikke? Desuden er der grænser for, hvor lange linjer må være. To skærmlinjer er maksimum.

Endte du med en irriterende lilla skærm? Kombinationen RUN STOP – RESTORE hjælper også her. Du får din »normale« skrivefarve tilbage samt en ren skærm. Men hverken dit program eller dine variabler har lidt overlast!

Vigtigst af alt: Du synes måske, programmet nu er blevet komplet uoverskueligt. Du vil imidlertid ved nærmere studium finde, at det er blevet det stik modsatte. Prøv at læse det:

1. Hvilken tabel?
2. Har vi været tabellen igennem?
3. Stil opgave, modtag svar, undersøg svar; hvis rigtigt, skriv RIGTIGT, tæl 1 rigtig!
4. Hvis forkert, skriv rigtigt svar!
5. Skriv, hvor mange rigtige!

Kan du det, har du taget det første skridt ind i programmeringens mysterier! Det er selvfølgelig ikke overvældende *spændende* programmer, vi har lavet indtil nu. Det vil vi prøve at rette på i næste lektion, hvor vi skal undersøge nogle af de matematiske funktioner, som vi vil finde kan bruges til andet og mere end matematik!

NB! Prøv

```
1 INPUT"TIM,MIN,SEK";T,M,S:PRINT"h"
2 PRINT"h"T"v "M"v "S"v ":S=S+1:IFS=60THENS=0:
  M=M+1
4 IFT=24THENT=0
5 P=P+1:IFP=103THENP=0:GOTO2
6 GOTO5
```

LEKTION 8

FUNKTIONER

12×34 er en OPERATION. Den knytter to tal sammen til et: 408. Men det er ikke alle operationer, der virker på to tal. Nogle virker kun på et. De kaldes FUNKTIONER.

Kvadratrod 4 er en sådan funktion. Kvadratroden af et tal er et andet tal, der ganget med sig selv giver det første. Kvadratrod 4 er 2, fordi $2 \times 2 = 4$. Det kan maskinen også regne ud. Prøv

?SQR(4)

og du får 2. Hvad er

?SQR(2)

Læg mærke til, at denne operation er en funktion, fordi den kun virker på ét tal, funktionens ARGUMENT. Dette argument er i *parenteser*. Nu er du måske ikke vild med at uddrage kvadratrødder, men så er der andre funktioner at lege med. Funktioner behøver nemlig ikke at virke på tal, de kan også virke på strenge. Tag

A\$="COMMODORE":?LEFT\$(A\$,3)

Vi får COM på skærmen.

LEFT\$(A\$,3)

betyder nemlig »de tre første karakterer af A\$«. Denne funktion vil vi vende tilbage til i en senere lektion. Der er andre, vi kan drage mere øjeblikkelig nytte af. Prøv

?RND(1)

Du får et tal mellem 0 og 1. Dette tal er *tilfældigt* (random). Kan du se nytten af dette i spil og lignende? Prøv igen. Du får et andet tilfældigt tal. Disse tal er altid mellem 0 og 1. Vi må regne lidt på dem for at gøre dem større og mere brugbare.

I virkeligheden er der selvfølgelig intet tilfældigt ved en datamat. Der er tale om en fast række tal, der er resultatet af så uigennemskuelige udregninger, at de forekommer tilfældige. Vi kan få maskinen til at starte et bestemt sted i følgen med

RND(-A)

hvor A er et eller andet tal. Prøv

?RND(1);RND(1);RND(1)

tre gange efter hinanden, og prøv tilsvarende

?RND(-27);RND(1);RND(1)

Er du med? Og så får vi i øvrigt det underlige E fra lektion 5 igen. Her har vi -08 i stedet for, et negativt tal, altså mindre end 0. Det betyder, at kommaet nu skal flyttes 8 pladser *til venstre*, så vi får altså i virkeligheden 0.0000000504123818. Din terning viste 0.0000000504123818 øjne!

Men når nu RND(1) giver et tal mellem 0 og 1, må RND(1)*6 da give et tal mellem 0 og 6. Nemlig, men noget helt tal er det stadig væk ikke. Det kan vi imidlertid lave det om til med en anden funktion:

INT(A)

betyder nemlig: Lav A om til et helt tal (smid decimalerne væk)!

`INT(27.2345)`

er 27. Og eftersom det tilfældige tal mellem 0 og 1, som maskinen skaber, aldrig er *helt* 1, giver

`?INT(RND(1)*6)`

os et tilfældigt helt tal mellem 0 og 6, og derfor kan vi bruge

`?INT(RND(1)*6)+1`

til at efterligne en terning. Et »endnu mere tilfældigt« tal kan vi få med

`RND(0)`

Denne version laver sit tilfældige tal efter et lille ur inde i maskinen, og det er derfor temmelig tilfældigt. Bad vi om det 10 eller 10 1/4 sekund over 5? Faktisk »går« maskinen med 1/60 sekunds nøjagtighed. Vi kan faktisk også bruge Commodore 64 som ur. Funktionen

`TI`

giver os antallet af tresindstyvendede af et sekund, der er gået, siden vi tændte for maskinen.

Bedre ville det selvfølgelig være, hvis vi kunne få tiden i timer, minutter og sekunder, og det kan vi faktisk med

`TI$`

Prøv

```
1 PRINT"h"TI$
2 RUN
```

Eller du kan »stille uret«. Er klokken 12.34.56, indtaster du blot

```
TI$="123456"
```

eller du kan nulstille det

```
TI$="000000"
```

og bruge det som stopur i spil med mere. Eller måske kunne du have lyst til at vide, hvor meget hukommelse du har i reserve. Tag

```
?FRE(0)
```

Får du et negativt tal, skal du lægge 65538 til.
Læg mærke til, at vi også kunne sige

```
?FRE(2345)
```

Her er tallet i parentesen ligegyldigt og er kun med for syntaksens skyld.

Der er selvfølgelig mange andre funktioner i COMMODORE 64, som vi skal udforske efterhånden. I denne lektion har vi kun opdaget netop så meget, at vi kan lave et spilleprogram med lidt mening i. Vi laver hele to i næste lektion.

NB! Prøv

```
1 PRINT"FIND MIT TAL (0-9)":A=INT(RND(1)*10)
2 INPUTB:C=C+1:IFB=ATHENPRINT"DU FIK DET
  I"C"FORSØG":END
3 PRINT"NEJ - PRØV IGEN":GOTO2
```

LEKTION 9

RELATIONER

For at forstå denne lektions to programmer er der endnu et fænomen, vi skal kende til: RELATIONEN.

Relationerne er

= LIG MED

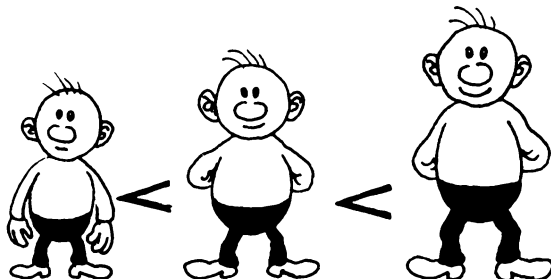
< > ULIG, FORSKELLIG FRA

> STØRRE END

< MINDRE END

>= STØRRE END ELLER LIG MED

<= MINDRE END ELLER LIG MED



Her kommer så det første program:

```
1 A=1000:B=10000
2 IFA=0THENPRINT"DU ER FALLIT":END
3 IFB<=0THENPPRINT"BANKEN ER FALLIT":END
4 PRINT"KAPITAL:"A:PRINT"f11f92f0f23f94f0f55f96f0
  f67f98f0f49"
```

```

5 INPUT" TAL";C:INPUT" INDSATS";D:
  IFD > A THEN 5
6 A=A-D:B=B+D:E=INT(RND(1)*10):
  PRINT"f2"E"KOM UD"
7 IFE=CTHENPRINT"DU VANDT":A=A+D*10:
  B=B-D*10:GOTO2
8 PRINT"f6DU TABTE":GOTO2

```

Linjer som 4 med mange farvekontroller skal man lige vænne sig til at læse, men så skulle der heller ikke være noget at tage fejl af: Farve 1: 1, farve 9 (RVS ON): 2, farve 0 (RVS OFF), farve 2: 3, osv. (lige tal er omvendte). Lad os igen prøve at »oversætte«:

1. Du skal have 1000 kr., og banken skal have 10000.
2. Hvis du har mistet alle dine penge, er du fallit, og spillet slutter.
3. Hvis banken har mistet alle sine penge, er den fallit, og spillet slutter.
4. Her er alt det, der skal stå på skærmen: Hvor mange penge, du har tilbage, de 9 tal, der kan spilles på.
5. Nu kan du indtaste, hvilket tal du holder på, og hvor meget. Hvis det er mere, end du har, bliver du spurgt en gang til.
6. Din indsats bliver trukket fra din kapital, banken får den, rouletten snurrer, og resultatet printes ud.
7. Hvis du har vundet, får du det at vide, du får dine penge ti gange igen, og et tilsvarende beløb bliver trukket fra bankens kapital, og du kan prøve en gang til.
8. Hvis du har tabt, får du det at vide, og du kan prøve en gang til.

Det var da ikke så svært, vel? Prøv selv at oversætte det følgende eksempel. m betyder mellemrum. Sæt selv farver til programmet.

```

1 PRINT"VI LEGER FIND TALLET. DIT TAL ER
  MELLEM 1 OG":INPUTA:B=A:PRINT"OKAY,"A
2 PRINT"HVIS DIT TAL ER STØRRE, TAST S;
  HVISMmmmmMINDRE, M; HVIS RIGTIGT, R"
3 C=1:D=INT((B-C)/2)+1
4 E=E+1:PRINTE". FORSØG:"D:IFB=CTHEN10
5 INPUTA$:IFA$="R"THEN10
6 IFA$="S"THENC=D+1:D=D+INT((B-D)/2)
7 IFA$="M"THENB=D-1:D=C+INT((D-C)/2)
8 IFB-D=1THEND=D+1
9 GOTO4
10 PRINT"JEG FANDT"D:PRINT"I"E"FOR SØG."
11 PRINT"KAN DU GØRE DET BEDRE? MIT TAL
  ER MELLEM1 OG"A
12 PRINT"OPGIVER DU, KAN DU TASTE 0":
  B=INT(RND(1)*A)+1
13 INPUTA:IFA=0THEN23
14 F=F+1:PRINTF". FORSØG:"A:IFA=BTHEN18
15 IFB > ATHENPRINT"S"
16 IFB < ATHENPRINT"M"
17 GOTO13
18 PRINT"DU FANDT MIT TAL":PRINT"I"F
  "FOR SØG.":PRINT"JEG FANDT DIT I"E"."
19 IFE < FTHENPRINT"JEG VANDT"
20 IFE > FTHENPRINT"DU VANDT"
21 IFE=FTHENPRINT"UAFGJORT"
22 END
23 PRINT"OKAY, DET VAR"B:PRINT"JEG VANDT"

```

Vi er nu tæt ved at være igennem begyndelsesgrundene i BASIC. Vi ved, hvad en variabel og et program er, kan bruge tastaturet og skrive primitive programmer med INPUT, IF og enkelte funktioner. I næste lektion skal vi fuldende vore forkundskaber med de logiske operationer AND, OR

og NOT. Her vil vi benytte lejligheden til at samle nogle småting op.

Vi brugte en del mellemrum i det sidste program. Faktisk kan sådanne overflødiggøres med funktionen

SPC(A)

hvor A er et tal. Prøv

?”COMMODORE 64”

og

?”COMMODORE”SPC(7)64

eller

?SPC(7)”COMMODORE”

I det sidste eksempel fik vi en lille »margen« på 7 pladser, næsten ligesom tabulatoren på en skrivemaskine. Faktisk har VIC en tabulatorfunktion, og

?TAB(7)”COMMODORE”

vil give det samme resultat som ovenstående.

Vi slutter af med et par FEJLMEDDELELSER, som du måske er stødt ind i. Prøv

```
1 INPUTA:IFA<0THEN3
2 PRINTSQR(A):END
4 PRINT”DUER IKKE”
```

Tast 4. Du får kvadratroden af 4, som er 2. Indtast nu -4. Man kan ikke tage kvadratroden af et negativt tal, derfor

har vi også en linje 4, der siger "DUER IKKE". Men 1 sender os til 3, som ikke eksisterer, så vi får

```
?UNDEF'D STATEMENT  
ERROR IN 1
```

UNDEFINED STATEMENT betyder udefineret ordre, det vil sige: der er ingen mening i, hvad du siger! men datamater har jo altid en høfligere måde at sige den slags sandheder på.

Hvad gør vi nu? Jo, vi kører naturligvis op og retter 3-tallet i linje 1 til et 4-tal. Så skulle maskinen faktisk kunne fortsætte:

```
CONT
```

Den svarer

```
?CAN'T CONTINUE  
ERROR
```

Det er jo et helt andet program, altså kan den ikke »fortsætte«!

NB! Prøv

```
1 PRINT"JEG FINDER DIT TAL(<=100), HVIS DU  
  GØR,SOM JEG SIGER. DEL DET"  
2 INPUT"MED 3! REST";A:INPUT"MED 5! REST"  
  ;B:INPUT"OG 7! REST";C:D=A*70+B*21+C*15  
3 IFD>105THEND=D-105:GOTO3  
4 PRINT"DIT TAL ER"D
```

LEKTION 10

LOGISKE OPERATIONER

Prøv

```
1 INPUT"STØRSTE PLANET";A$:IFA$="JUPITER"  
  THENEND  
2 RUN
```

Programmet stiller opgaven, til det rigtige svar indtastes.
Prøv nu

```
1 INPUT"MARS' DRABANTER";A$,B$:IFA$="PHO  
  BOS"ANDB$="DEIMOS"THENEND  
2 RUN
```

Nu standser programmet kun, hvis *begge* svar er rigtige.
Men sæt nu, eleven svarer i omvendt rækkefølge? Så må vi have

```
1 INPUT"MARS' DRABANTER";A$,B$  
2 IFA$="PHOBOS"ANDB$="DEIMOS"ORA$=  
  "DEIMOS"ANDB$="PHOBOS"THENEND  
3 RUN
```

OR betyder eller. Hvis blot en af betingelserne er opfyldt, standser programmet, altså enten

1. A\$="PHOBOS" og B\$="DEIMOS

eller

2. A\$="DEIMOS" og B\$="PHOBOS"

Men hvordan kan maskinen nu vide, at vi ikke mener, at programmet skal stoppe, hvis det er opfyldt, at

1. A\$="PHOBOS"

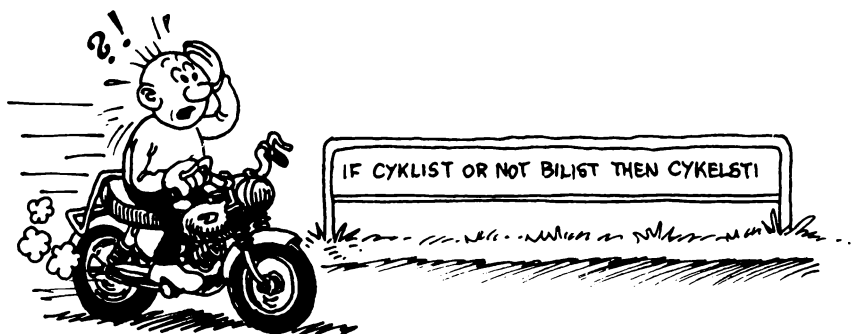
og

2. B\$="DEIMOS" eller A\$="DEIMOS"

og

3. B\$="PHOBOS"

Dette ville jo aldrig ske, for hvis A\$ og B\$ begge er "PHOBOS", kan ingen af dem være "DEIMOS". Svaret er igen »prioritering«. Maskinen tager simpelt hen alle AND, før den begynder med OR. Der er imidlertid en logisk operation, den udfører før både AND og OR, og det er NOT: ikke.



Lad os igen se på lektionens første eksempel. Faktisk kunne vi have nøjedes med en enkelt linje:

```
1 INPUT"STØRSTE PLANET";A$:IFNOTA$="JUPITER"THENRUN
```

Nu siger programmet i stedet for, at maskinen skal spørge igen, hvis svaret *ikke* er rigtigt, og det kommer jo faktisk ud på et. Kunne vi gøre det samme med det næste eksempel:

```
1 INPUT"MARS' DRABANTER";A$,B$:IFNOT(A$="PHOBOS"ANDB$="DEIMOS")THENRUN
```

Hvad skal vi med parenteser? Ja, hvad ville

```
1 INPUT"MARS' DRABANTER";A$,B$:IFNOTA$="PHOBOS"ANDB$="DEIMOS"THENRUN
```

betyde? Nemlig: Spørg igen, hvis det er opfyldt, at

1. A\$ ikke er "PHOBOS"

og

2. B\$ er "DEIMOS"

Maskinen tager NOT før AND, ikke? Måske synes du, linjen ser lidt mystisk ud. Kunne vi ikke omskrive den på en måde, så vi kan undvære parenteser? Hvad med

```
1 INPUT"MARS' DRABANTER";A$,B$:IFNOTA$="PHOBOS"ANDNOTB$="DEIMOS"THENRUN
```

Det ser da plausibelt ud, ikke? Men pas nu på: Det betyder jo, at *både* PHOBOS og DEIMOS skal være forkerte, hvis ma-

skinen skal gå tilbage. Og da vi jo må forlange begge rigtige, for at maskinen skal stoppe, skal også den ene forkert være nok til at sende maskinen tilbage.

```
1 INPUT"MARSDRABANTER";A$,B$:IFNOT(A$="PHOBOS"ANDB$="DEIMOS")THENRUN
```

må altså oversættes ved

```
1 INPUT"MARSDRABANTER";A$,B$:IFNOTA$="PHOBOS"ORNOTB$="DEIMOS"THENRUN
```

Hvad nu, hvis linjen havde heddet

```
1 INPUT"MARSDRABANTER";A$,B$:IFNOT(A$="PHOBOS"ORB$="DEIMOS")THENRUN
```

Hvornår ville maskinen så være gået tilbage? *Når det ikke var tilfældet, at*

(A\$="PHOBOS"ORB\$="DEIMOS"). Med andre ord, når *ingen af delene* var tilfældet. Dette udtrykker altså i virkeligheden et »hverken-eller« og kan oversættes

```
1 INPUT"MARSDRABANTER";A$,B$:IFNOTA$="PHOBOS"ANDNOTB$="DEIMOS"THENRUN
```

Vi kan også udtrykke os lidt mere generelt, idet vi lader bogstaverne P og Q (en slags variabler) stå for hele relationer. Så kan vi skrive

NOT(PANDQ) er det samme som NOTPORNQ
NOT(PORQ) er det samme som NOTPANDNOTQ

Husk endelig på, at

NOTA=B

er præcis det samme som

$A < > B$

På samme måde er $A > B$ det modsatte af $A \leq B$ og $A < B$ det modsatte af $A \geq B$.

Vi er nu som sagt gennem begyndelsesgrundene i BASIC. I næste del skal vi udforske sprogets muligheder nærmere.

NB! Prøv

```
1 A=25:PRINT" NIM"
2 PRINT"DER ER" A "TILBAGE":IFA < 2THENPRINT"
  JEG VANDT":END
3 INPUT"HVOR MANGE TAGER DU";B:IFB
  < 1ORB > 3THEN3
4 C=4-B:PRINT"JEG TAGER"C:A=A-B-C:GOTO2
```


ANDEN DEL
Data og sæt

LEKTION 11

FOR-NEXT-LØKKER

Prøv

```
10 PRINT"COMMODORE 64"  
20 GOTO10
```

Som vi tidligere har set, er dette en løkke, der skriver Commodore 64 på skærmen, til vi taster RUN STOP. Det er en såkaldt »uendelig« og komplet ukontrollabel løkke. Vi kan imidlertid kontrollere den med

```
10 PRINT"COMMODORE 64"  
20 A=A+1:IF A < 10THEN10
```

Dette kan vi imidlertid også udtrykke på en anden måde:

```
5 FORA=1TO10  
10 PRINT"COMMODORE 64"  
20 NEXTA
```

Fænomenet kaldes en FOR-NEXT-LØKKE. A er en KONTROLVARIABEL. Den svarer til tælleren i det første eksempel, idet den gennemløber de værdier, FOR-linjen definerer, og således sørger for, at alt mellem FOR og NEXT bliver udført et bestemt antal gange. Vi kan sagtens PRINTe kontrolvariablen ud, ligesom vi kan have flere FOR-NEXT-løkker inde i hinanden:

```
1 FORA=1TO10:FORB=1TO10:PRINTA*B:NEXTB:NEXTA
```

skriver den lille tabel ud. Læg mærke til, at løkkerne skal ligge inden i hinanden, så maskinen ikke støder på en »forkert« NEXT. Et andet legalt eksempel ville være

```
FORA
FORB
FORC
NEXTC
NEXTB
FORD
NEXTD
NEXTA
```



Ofte kan vi undvære at nævne kontrolvariablen efter NEXT. Så vil maskinen nemlig automatisk opfatte det som hørende til den sidst påbegyndte løkke.

Prøv

```
1 FORA=1TO10:PRINT"COMMODORE 64"
```

og

```
1 PRINT"COMMODORE 64":NEXT
```

Prøv nu

```
1 FORA=1TO10STEP.5:PRINTA:NEXT
```

Nu tæller kontrolvariablen i trin af .5 (1/2). Vi kan også have negativt STEP:

```
1 FORA=10TO0STEP-1:PRINTA:NEXT
```

Kontrolvariablen giver os faktisk en ret enestående kontrol. Tag f.eks. vores tabelprogram, hvor tallene bare farer af sted. Vi kunne nu have

```
1 FORA=1TO10:FORB=1TO10:PRINTA*B:NEXT:INP  
UT:NEXT
```

eller

```
1 FORA=1TO10:FORB=1TO10:PRINTA*B:NEXT:FO  
RC=1TO5000:NEXT:NEXT
```

Læg mærke til, at C-løkken ingenting gør. Den er bare en pause. Hvor lang er den? Prøv

```
1 TI$="000000":FORA=1TO10000:NEXT:PRINTTI/60
```

Maskinen »tæller til« 10000 på 11,2 sekunder. En datamats kvalitet kan ofte ses på dens hurtighed, og COMMODORE 64 er faktisk temmelig hurtig i forhold til sine konkurrenter, TEXAS, COLOR GENIE, SPECTRUM, osv. Allerlangsomst (men også allerbilligst) er selvfølgelig ZX81, der ville være 3 minutter og 20 sekunder om ovenstående løkke. SPECTRUM ville kunne klare den på 50 sekunder.

Lidt forenklet kan vi sige, at skal vi have en pause på 1 sekund, skal FOR-NEXT-løkken på 64 gå til 1000. Det kan måske synes lidt overflødigt med en så stor hurtighed i udregningsprogrammer, men vi bliver glade for den, når vi skal til at lave TV-spil!

Eller prøv

```
1 FORA=1TO22:FORB=1TO40:PRINT"flS";:NEXT:NE  
XT
```

Vi får 20 rækker hjerter. Prøv at skifte As højeste værdi (dens »grænseværdi« – 1 er »startværdien«) ud med 23. Vi fik ikke nogen ekstra linje, tværtimod.

Maskinen skaffer plads til sit READY.

Prøv nu

```

1 PRINT"fl":FORA=1TO25:FORB=1TO40:IFA=25AND
  DB=40THENPRINT"vi";
2 PRINT"S";:NEXT:NEXT
3 GOTO3

```

i står for INSERT. Når du kommer til denne karakter, holder du bare SHIFT nede og taster INST. Kan du se virkningen? Maskinen skifter linje, når vi har skrevet på dens sidste plads. Men i dette program skriver vi *aldrig* på den sidste plads, vi skriver på den *næstsidste*, og så går vi et skridt tilbage (*v = markør til venstre*) og laver et hul (INSERT), før vi skriver den sidste karakter. Endelig »fryser« vi billedet med 3. 2 kunne også have heddet

```

2 PRINT"S";:NEXTB,A

```

De to linjer er ens, kun udtryksmåden er forskellig. Læg også mærke til, at de mange 1-linjes-eksempler sagtens kunne undvære linjenummer og indtastes som direkte ordrer, selv om de i virkeligheden er sammensat af flere sådanne! Prøv til slut

```

1 INPUT"PRIMTAL MELLEM";A:INPUT"OG";B:
  IFA <=2THENA=2:PRINT2;;C=1
2 FORD=ATO B:FORE=2TOSQR(D):IFD/E=INT(D/E)
  THEN4
3 NEXTE:PRINTD;;C=C+1
4 NEXTD:PRINTC"PRIMTAL"

```

NB! Prøv

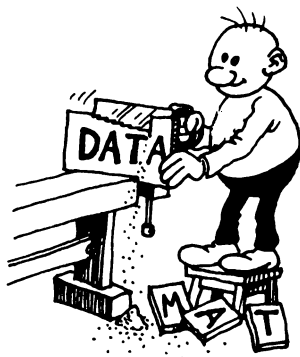
```

1 A=1: INPUTB: FORC=1TOB:A=A*C:NEXT:PRINTB"v!
  (FAKULTET)=":PRINTA:RUN

```

LEKTION 12

STRENGDELING



Kan du huske

```
1 INPUT "MARS' DRABANTER";A$,B$
2 IFA$="PHOBOS"ANDB$="DEIMOS"ORA$="DEIMOS"
  S"ANDB$="PHOBOS"THENEND
3 RUN
```

Den lange linje skyldtes jo, at vi ikke kunne vide, om eleven ville svare PHOBOS eller DEIMOS *først*. Men tag så f.eks.

```
1 INPUT "STØRSTE FUGL";A$:IFA$ < > "STRUDS"
  THENRUN
```

Sæt nu, der svares EN STRUDS eller STRUDSEN eller STRUDSE? Ja, så er svaret »forkert«. Det ville kræve alt for mange ANDer at dække alle mulige korrekte besvarelser. Problemet er jo, at maskinen ikke virkelig forstår, hvad vi siger.

Hvordan forstår vi da selv svar som ovenstående som rigtige? Jo, vi ser efter, om de indeholder STRUDS. Det kan maskinen også gøre. Til det brug har den nemlig en række funktioner:

LEN(A\$) betyder antallet af karakterer i strengen A\$
LEFT\$(A\$,A) betyder de A første karakterer i strengen A\$
RIGHT\$(A\$,A) betyder de A sidste karakterer i strengen A\$
MID\$(A\$,A,B) betyder B karakterer fra strengen A\$ begyndende med karakter nummer A

I det sidste tilfælde kan vi underforstå B. Så betyder

MID\$(A\$,A) strengen A\$ fra og med karakter nummer A, og strengen ud.

Læg mærke til, at

MID\$(A\$,A,1)

åbenbart betyder: Karakter nummer A i strengen A\$! Lad os prøve funktionerne på strengen

A\$="KØBENHAVN"

Vi får nu f.eks.

LEN(A\$)=9

LEFT\$(A\$,3)="KØB"

RIGHT\$(A\$,4)="HAVN"

MID\$(A\$,4,2)="EN"

MID\$(A\$,4)="ENHAVN"

MID\$(A\$,4,1)="E"

Eller prøv

?LEFT\$(A\$,5)+RIGHT\$(A\$,3)

```
1 A$="KØBENHAVN":FORA=1TO8:PRINTMID$(A$,  
  A,2):NEXT
```


skriver alle mulige DELSTRENGE af A\$ på 2 karakterer ud.
Laver vi dette program om til

```
1 A$="KØBENHAVN":FORA=1TO8:IFMID$(A$,A,2)=  
  "EN"THENPRINT"OK"  
2 NEXT
```

får vi et program, der checker, om en af disse delstreng er
"EN". Vi kan nu lave vores strudse-program om til

```
1 INPUT"STØRSTE FUGL";A$:FORA=1TOLEN(A$):IF  
  MID$(A$,A,6)="STRUDS"THENEND  
2 NEXT:RUN
```

Vi kan også gå den anden vej og lave et program, der kan
konstatere forekomsten af forskellige delstreng i strengen
"KØBENHAVN". Prøv

```
1 A$="KØBENHAVN":INPUT"DELSTRENG";B$:FORA  
  =1TO9:IFMID$(A$,A,LEN(B$))=B$THEN?"OK"  
2 NEXT
```

eller helt generelt

```
1 INPUT"STRENG";A$:INPUT"DELSTRENG";B$  
2 FORA=1TOLEN(A$):IFMID$(A$,A,LEN(B$))=B$THEN  
  PRINT"OK"  
3 NEXT
```

Hvad gør

```
1 INPUTA$:FORA=1TOLEN(A$):IFMID$(A$,A,1)="A"  
  THENB=B+1  
2 NEXT:PRINTB
```

Eller prøv

```
1 INPUT"TEKST";A$:FORA=1TOLEN(A$)
2 IFMID$(A$,A,8)="COMPUTER"THEN A$=LEFT$(A$,
  A-1)+"DATAMAT"+RIGHT$(A$,LEN(A$)-A-7)
3 NEXT:PRINTA$
```

Programmet undersøger A\$ for forekomsten af delstren-
gen COMPUTER og skifter den ud, hvor den forekommer,
med det mere danske DATAMAT. Generelt kan vi have

```
1 INPUT"TEKST";A$:INPUT"RET";B$:INPUT"TIL";C$
  :FORA=1TOLEN(A$)
2 IFMID$(A$,A,LEN(B$))=B$THEN A$=LEFT$(A$,A-1)
  +C$+RIGHT$(A$,LEN(A$)-A-LEN(B$)+1)
3 NEXT:PRINTA$
```

Prøv

```
1 PRINTMID$("f1f2f3f4f5f6f7f8c1c2c3c4c5c6c7c8",
  INT(RND(1)*16)+1,1)"f9m";:RUN
```

og

```
1 INPUT"TEKST";A$:FORA=1TO40-LEN(A$):A$=A$+
  "m":NEXT:PRINT"h"
2 PRINT"hf2"LEFT$(A$,40)
3 A$=RIGHT$(A$,LEN(A$)-1)+LEFT$(A$,1):FORA=1T
  O100:NEXT:GOTO2
```

NB! Prøv til sidst

```
1 INPUT"ORD";A$:FORA=LEN(A$)TO1STEP-1: B$ =
  B$ + MID$(A$,A,1):NEXT:A$=B$:PRINTA$:RUN
```

LEKTION 13

GOSUB

Som vi har set, giver

```
1 A$="KØBENHAVN":PRINTMID$(A$,3,1)
```

os bogstavet B. Kunne vi ikke på samme måde bede om det tredje *ciffer* i f.eks. tallet 123456789? Vi prøver

```
1 A=123456789:PRINTMID$(A,3,1)
```

Vi får TYPE MISMATCH (se FØRSTE DEL LEKTION 6), eftersom vi forsøger at bruge en funktion, der kun egner sig til strenge, på et tal. Det, vi skal bruge, er en funktion, der laver et tal om til en streng. Den har vi i

STR\$

Prøv

```
1 A=123456789:A$=STR$(A):PRINTMID$(A$,3,1)
```

Vi får 2! Ja, for strengen A\$ er nemlig på 10 karakterer, hvad vi kan overbevise os om med LEN(A\$). Den første karakter er et mellemrum, der i virkeligheden er et underforstået +, hvilket betyder, at A er et positivt tal. Sæt, vi nu vil have vores 2-tal ganget med 3? Vi forsøger

```
1 A=123456789:A$=STR$(A):PRINTMID$(A$,3,1)*3
```

Mere miskmask. Nu prøver vi gud hjælpe mig at gange en streng med 3. Nu skal vi bruge

VAL

der laver en streng om til et tal. Vi kan så have

```
1 A=123456789:A$=STR$(A):B=VAL(MID$(A$,3,1)):PR
INTB*3
```

Prøv

```
1 A$="f1f2f3f4f5f6f7f8":FORA=1TO4:B=B+(INT(RN
D(1)*8)+1)*10p(6-A):NEXT
2 B$=MID$(STR$(B),2,4)
3 C=C+1:PRINTC;:PRINT"v. FORSØG";:IFC < 10TH
ENPRINT"ø";
4 INPUTC$?: "nøøøøøøøøøøøøøøøø";:FORA=1TO4:
MID$(A$,VAL(MID$(C$,A,1)),1)"Q";:NEXT
5 D=0:E=0:FORA=1TO4:IFMID$(C$,A,1)=MID$(B$,A,
1)THEND=D+1:GOTO8
6 FORB=1TO4:IFMID$(C$,A,1)=MID$(B$,B,1)THENE=
E+1:GOTO8
7 NEXTB
8 NEXTA:PRINT"f1";:IFD=0THEN10
9 FORA=1TOD:PRINT"*";:NEXT
10 IFE=0THEN12
11 FORA=1TOE:PRINT".";:NEXT
12 IFC$=B$THENEND
13 PRINT:GOTO3
```

Frit oversat:

1. Maskinen tænker på et tal på fire cifre mellem 1 og 8.
2. Ditto.

3. Maskinen regner ud, hvor mange forsøg du har brugt.
4. Maskinen tager imod dit gæt og skriver det på skærmen som farver. Tolket på denne måde er det en farvekom-
bination, den har tænkt på, og som du skal gætte ved at
indtaste tallene på farvetasterne. Er du med?
5. Maskinen undersøger, om du har placeret nogen farver
korrekt.
6. Maskinen undersøger, om du har placeret en rigtig far-
ve et forkert sted.
7. Slut på B-løkken.
8. Slut på A-løkken. Skift til »normal« farve. Hvis ingen far-
ver er placeret korrekt, videre til 10!
9. Maskinen skriver, hvor mange farver korrekt placeret.
10. Hvis ingen rigtige farver placeret forkert, videre til 12!
11. Maskinen skriver, hvor mange rigtige farver forkert pla-
ceret.
12. Hvis alle rigtige, slut!
13. Ny linje. Om igen, soldat!

Prøv også

```

1 A$="HVAD ER HOVEDSTADEN I":PRINTA$:INPUT
  T"DANMARK";B$:C$="KØBENHAVN":GOSUB7
2 PRINTA$:INPUT"SVERIGE";B$:C$="STOCKHOLM":
  GOSUB7
3 PRINTA$:INPUT"NORGE";B$:C$="OSLO":GOSUB7
4 PRINTA$:INPUT"FRANKRIG";B$:C$="PARIS":GOS
  UB7
5 PRINTA$:INPUT"ITALIEN";B$:C$="ROM":GOSUB7
6 PRINTA$:INPUT"SPANIEN";B$:C$="MADRID":GOS
  UB7:END
7 FORA=1TOLEN(B$):IFMID$(B$,A,LEN(C$))=C$THEN
  PRINT"RIGTIGT":RETURN
8 NEXT:PRINT"NEJ - "C$:RETURN

```

Det er igen strengdelingen, der er på spil. Men i stedet for at sætte denne strengdeling efter hvert spørgsmål, findes den kun ét sted i programmet, nemlig linje 7 og 8. Det lille ord GOSUB sender ligesom GOTO maskinen til denne linje for hvert spørgsmål, men det særlige ved GOSUB er, at den selv finder »hjem« igen. RETURN betyder »gå videre i programmet, hvor du slap«.

Læg mærke til END i linje 6. Prøv at slette det. Vi får

```
?RETURN WITHOUT GOSUB  
ERROR IN 7
```

eller måske (hvis du svarer forkert) ERROR IN 8. Jo, se, da maskinen havde klaret det sidste spørgsmål, fortsatte den i 7 og stødte derfor på et RETURN uden GOSUB. Den var jo nemlig ikke blevet sendt ind i denne SUBROUTINE af et GOSUB, men var af vanvare vadet derind »efter kampen«.

Læg også mærke til, at vi ikke behøver at skrive HVAD ER HOVEDSTADEN I for hvert spørgsmål, eftersom vi har oplagret denne (faste) del af spørgsmålet i A\$.

Selvfølgelig skal vi stadig bruge en lang linje til hvert enkelt spørgsmål. Tænk, hvis vi i stedet kunne proppe det hele ind i en enkelt FOR-NEXT-løkke: Spørgsmål 1, svar 1, osv. Det kan vi faktisk godt. Men så skal vi først vide noget om SÆT!

NB! Prøv

```
1 INPUT "TAL";A:A$=STR$(A):FORA=2TOLEN(A$):  
  B=B+VAL(MID$(A$,A,1)):NEXT:PRINTB:RUN
```

LEKTION 14

SÆT

Et sæt er et system af variabler, hvis navne kun adskiller sig numerisk fra hinanden, hvilket giver os mulighed for at behandle dem helt anderledes effektivt.

```
1 FORA=1TO10:A(A)=37:NEXT
```

tilskriver på en og samme gang værdien 37 til ti forskellige variabler, som vi senere kan bruge som $A(1), A(2), \dots A(10)$. Men prøv så at køre

```
1 FORA=1TO100:A(A)=37:NEXT
```

Vi får

```
?BAD SUBSCRIPT  
ERROR IN 1
```

Fejlmeddelelsen betyder »forkert indeks«. Indeks er tallet i parentes. Det er klart, at vi på én gang forlanger en forfærdelig masse plads af maskinen, når vi i en enkelt linje vil tilskrive 100 variabler. Skal vi derfor danne sæt på over 11 INDICEREDE VARIABLER, må vi først gøre maskinen opmærksom på, at vi kommer til at behøve denne plads. Det gør vi med ordren

```
DIM
```

Vi kan så have

```
1 DIMA(100):FORA=1TO100:A(A)=37:NEXT
```

og checke med

```
?A(77)
```

Det fungerer. Faktisk har vi med DIMENSIONERINGS-ORDREN skaffet plads til 101 variabler, idet den første variabel i et sæt altid har indeks 0! Vi kan også dimensionere STRENGSÆT. Vi kunne altså sagtens have haft

```
1 DIMA$(100):FORA=1TO100:A$(A)="KØBENHAVN":  
NEXT
```

Endelig kan vi danne sæt i flere dimensioner. Prøv

```
DIMB(15,15)
```

Vi har nu $16 \cdot 16 = 256$ variabler til rådighed med navnene $B(0,0), B(0,1), B(0,2), \dots B(0,15), B(1,0), \dots B(15,15)$

Prøv nu

```
DIMA(15)
```

Vi får

```
?REDIM'D ARRAY  
ERROR
```

På denne måde forhindrer maskinen os i at slette et gammelt sæt med et nyt ved en fejltagelse.

En mand ved navn Conway opstillede engang et sæt (temmelig forenklede) leveregler for levende celler. Han gik

ud fra en »flad« koloni, som den kunne brede sig over en glasplade, altså

```
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
```

Som det vil fremgå, har hver celle, bortset fra de yderste, 8 naboer – dette er selvfølgelig efter dette system det største antal naboer, en celle kan have. I kolonien

```
0
00
```

har hver celle 2 naboer. CONWAYS REGLER siger nu:

1. En celle med 2 eller 3 naboer overlever til næste generation.
2. En celle »fødes« i *næste generation* i et felt, der er tomt i *denne generation*, hvis dette tomme felt har nøjagtig 3 naboer.

Underforstået i 1. er, at en celle med *under 2* eller *over 3* naboer dør. Dette skal vi nu forsøge at lave om til et program: LIFE. Vi prøver

```
1 DIMA(25,42),B(25,42):A=-1:FORB=2TO24:FORC
  =2TO41:IFRND(1)<.5THENB(B,C)=1
```

```

2 NEXT:NEXT:PRINT"h"
3 A=A+1:PRINT"hf4f9GENERATION"A"f1":D=0:FOR
  B=2TO24:FORC=2TO41:A(B,C)=B(B,C)
4 IFA(B,C)=1THENPRINT"Q";:D=D+1
5 IFA(B,C)=0THENPRINT"m";
6 NEXT:NEXT:PRINT"f4f9POPULATION"D"vm";:FO
  RB=2TO24:FORC=2TO41:D=0:D=D+A(B-1,C-1)
7 D=D+A(B-1,C)+A(B-1,C+1)+A(B,C-1)+A(B,C+1)+
  A(B+1,C-1)+A(B+1,C)+A(B+1,C+1)
8 IFA(B,C)=1ANDD < > 2ANDD < > 3THENB(B,C)
  =0
9 IFA(B,C)=0ANDD=3THENB(B,C)=1
10 NEXT:NEXT:GOTO3

```

Og så er vi parate til en lille studie i cellevækst. Jeg bliver aldrig træt af dette simple program; at sidde og betragte de besynderlige livsformer, der udvikler sig spontant, udelukkende for at overleve under de givne naturlove. Hvordan de indvirker på hinanden og endelig stabiliseres i perfekte former eller dør.

NB! Prøv

```

1 INPUT"ANTAL TERNINGER";A:INPUT"ANTAL
  KAST";B:DIMA(A*6)
2 FORC=1TOB:FORD=1TOA:E=E+INT(RND(1)*6)+1:
  NEXT:A(E)=A(E)+1:E=0:NEXT
3 FORC=1TOA*6:PRINTC": "A(C)*100/B "%":NEXT

```

LEKTION 15

DATA

I lektion 14 brugte vi et NUMERISK SÆT I 2 DIMENSIONER til at fremstille et program, der simulerede en cellekolonis liv og død. I denne lektion skal vi se lidt på, hvad vi kan bruge STRENGSÆT til. Prøv

```
1 DIMA$(18):FORA=1TO18:READA$(A):NEXT
2 INPUTA$:FORA=1TO18:FORB=1TOLEN(A$(A)):IF
  MID$(A$(A),B,LEN(A$))=A$THENPRINTA$(A)
3 NEXT:NEXT:GOTO2
4 DATAKLAUS RIFBJERG*JUS OG/ELLER DE
  N GYLDNE MIDDELVEJ
5 DATAEGON NIELSEN*SALAMANDERDAGE,AN
  DERS BODELSEN*OVER REGNBUE
6 DATAERIK WEDERSØE*18 TONS DRØMME,DE
  A TRIER MØRCH*AFTENSTJERNEN
7 DATADORRIT WILLUMSEN*NI LIV,FINN ABRA
  HAMOWITZ*SKYGGEN FØR MØRKET
8 DATAVAGN LUNDBYE*JONAS TRILOGIEN,CH
  ARLOTTE STRANDGAARD*LILLE MENNESKE
9 DATAANNA LADEGAARD*REJSE MED MIN SØ
  N,ANGELO HJORT*DEN SOVENDE BY
10 DATAHANS LYNGBY JEPSEN*SOMMER I SEPT
  EMBER,LENE H. BAGGER*IDIOTERNE
11 DATAOLE FRØSTRUP*YETIERNES SANG,POUL-
  HENRIK TRAMPE*FORHOLD
12 DATAERWIN NEUTZSKY-WULF*MENNESKE,
  HARLY FOGED*SKABET
```

13 DATAKNUD H. THOMSEN*RØVERNE I SKOTLAND

Når vi kører dette program, viser det os et spørgsmålstegn: Signal til INPUT. Vi kan nu indtaste et SØGEORD, og programmet vil ud fra dette finde den rigtige bog. TRAMPE vil altså føre os frem til POUL HENRIK TRAMPE*FORHOLD, og JONAS til Vagn Lundbye. Dette er ærketypen på en database, et datakartotek, der er alle kartoteker på én gang; efter forfatter, titel, eller hvad vi måtte ønske. Men hvordan virker det? Vi kan umiddelbart inddele ovenstående program i tre underprogrammer:

1. Initialisering.
- 2.-3. Analyse.
- 4.-13. Data.

Analysen er selvfølgelig programmets egentlige mekanisme, og den er faktisk ret banal strengdeling. Men vi skal jo også have vore oplysninger ind i sættet, og det er denne proces, vi hentyder til som initialisering. Vi kunne have haft

```
1 DIMA$(20):A$(1)="KLAUS RIFBJERG*JUS OG/ELL  
ER DEN GYLDNE MIDDELVEJ"
```

osv., men det havde ikke været særlig praktisk. En måde er

```
1 DIMA$(20):FORA=1TO20:INPUT$(A):NEXT
```

Nu »beder« maskinen simpelt hen om de 20 indtastninger og gemmer dem i sættet. Programmet er selvfølgelig lettere at overse, hvis sættets værdier står at læse i det, og jeg har derfor her valgt en tredje metode, en serie DATA-sætninger uden for det egentlige program. Formlen

READA\$(A)

betyder nu, at maskinen skal læse DATA ind i sættet en for en. Til terminologien hører også RESTORE, der betyder, at maskinen skal begynde forfra på DATA.

Tag

```
1 DIMA(100):FORA=1TO100:READA(A):IFA/7=INT(A/  
7)THENRESTORE  
2 NEXT  
3 FORA=1TO100:PRINTA(A);:NEXT  
4 DATA1,2,3,4,5,6,7,8,9,10
```

Læg mærke til, at vi ikke behøver gåseøjne om DATA, med mindre udeladelsen af dem kan give anledning til misforståelser, som hvis vi havde haft et , med i en af bogtitlerne. Faktisk brugte vi * i stedet for : mellem forfatter og titel, eftersom : uden "" ville have betydet noget i retning af »slut på data«.

Prøv på denne måde at skære data fra i det sidste program, f.eks. efter 5. Maskinen klager

?OUT OF DATA
ERROR IN 1

Metoden overflødiggør som sagt alfabetiske kartoteker, men ønsker vi netop alfabetisering, kan vi også få det:

```
1 DIMA$(100):A$(1)="ZZ"  
2 A=A+1  
3 INPUTA$:IFA$=":"THEN7  
4 FORB=1TO100:IFA$<A$(B)THEN6  
5 NEXT  
6 FORC=ATO BSTEP-1:A$(C+1)=A$(C):  
NEXT:A$(B)=A$:GOTO2
```

7 FORB=1TOA-1:PRINTA\$(B):NEXT:GOTO3

Når du ønsker at se den (foreløbige) alfabetiserede liste, maskinen har lavet ud af dine indtastninger, skal du blot taste *. Så kommer listen frem, hvorpå maskinen er parat til at modtage yderligere indtastninger.

Princippet i det hele er maskinens behagelige evne til at vurdere strenge ved at lade dem indgå i relationer.

A\$ < B\$

betyder altså simpelt hen: A\$ kommer før B\$ i alfabetet. Hvert bogstav har nemlig en numerisk værdi, den såkaldte ASCII-kode, som vi senere skal vende tilbage til.

Læg også mærke til, hvordan programmet hele tiden laver et »hul« til det nye ord ved at skubbe alle de tidligere en tak!

I 16. lektion skal vi bl.a. se på et program, der er i stand til at oversætte en given tekst fra ét sprog til et andet!

NB! Prøv

```
1 A=A+1:READA$(A):IFA$(A) < > "SLUT"THEN1
2 FORA=1TO9:PRINTA,A$(A):NEXT
3 DATAET,TO,TRE,FIRE,FEM,SEKS,SYV,OTTE,NIS,
  LUT
```

LEKTION 16

STRENGSÆT

```
1 DIMA$(77),B$(77),C$(25):FORA=1TO77:READA$(A),  
  B$(A):NEXT  
2 INPUTA$:A$=A$+" ":A=2:B=1:C=1  
3 IFMID$(A$,A,1)=" "THENC$(B)=MID$(A$,C,A-C):B  
  =B+1:C=A+1  
4 IFA=LEN(A$)THEN6  
5 A=A+1:GOTO3  
6 FORA=1TOB-1:FORC=1TO77:IFC$(A)=A$(C)THEN  
  C$(A)=B$(C):GOTO8  
7 NEXTC  
8 PRINTC$(A)" ";:NEXTA  
9 DATACOME,KOMME,GET,SKAFFE,GO,REJSE,KEE  
  P,HOLDE,LET,LADE,MAKE,LAVE,PUT,ANBRINGE  
10 DATASEEM,FOREKOMME,TAKE,TAGE,BE,BLIVE,  
  DO,GØRE,SAY,SIGE,SEE,SE,SEND,SENDE  
11 DATAWILL,VIL,ABOUT,OMKRING,ACROSS,OVE  
  R,AFTER,EFTER,AGAINST,MOD,AMONG,IBLANDT  
12 DATABEFORE,FØR,BETWEEN,IMELLEM,BY,AF,D  
  OWN,NED,FROM,FRA,IN,I,OFF,BORT  
13 DATATO,TIL,UP,OP,WITH,MED,AS,LIGESOM,OF,  
  AF,TILL,TIL,THAN,END,A,EN,THE,DEN  
14 DATAANY,NOGEN,EVERY,ENHVER,NO,NEJ,OTH  
  ER,ANDEN,SOME,NOGEN,THAT,DEN,I,JEG  
15 DATAHE,HAN,YOU,DU,WHO,HVEM,AND,OG,BE  
  CAUSE,FORDI,BUT,MEN,OR,ELLER,IF,HVIS  
16 DATATHOUGH,SKØNT,WHILE,MEDENS,HOW,H  
  VORDAN,WHEN,DA,WHERE,HVOR,WHY,HVOR  
  FOR
```

- 17 DATAFAR,FJERN,FORWARD,FREMADE,HERE,HER,
NOW,NU,OUT,UD,THEN,DA,THERE,DER
18 DATATOGETHER,SAMMEN,WELL,GODT,ENOUGH,
NOK,EVEN,ENDO,SMALL,LITTLE,LILLE,NOT,IKKE
19 DATAONLY,KUN,QUITE,HELT,VERY,MEGET,NORTH,
NORD,SOUTH,SYD,EAST,ØST,WEST,VEST

Prøv nu programmet med originale og intelligente sætninger som

IF YOU GO WEST THEN I WILL COME WITH
YOU

eller

I WILL NOT LET YOU GO THERE

Måske synes du ikke umiddelbart, at

HVIS DU REJSE VEST DA JEG VIL KOMME MED
DU

eller

JEG VIL IKKE LADE DU REJSE DER

er synderlig gode oversættelser, men tænk så på, hvis der var tale om en russisk tekst, som du ikke forstod et eneste ord af. Så ville en oversættelse af denne type ikke være så ringe. Selvfølgelig er programmet ikke meget værd i den form, det har nu. 77 gloser er ikke meget, og ikke et eneste navneord! Men vi kan da også sagtens få plads til flere. De 77 gloser med oversættelse fylder knap en K. 3000 gloser skulle altså være inden for mulighedernes grænse. Lidt grammatik kunne man måske også indlægge i program-

met, så det blev lidt mindre undersættelse. I sig selv er programmet enkelt:

1. Tag imod gløser og den tekst, der skal oversættes.
2. Del tekst op i enkelte ord.
3. Mere tekst?
4. Videre i tekst!
5. Oversæt et ord ad gangen.
6. Videre i data!
7. Skriv oversættelse.
- 8.-18. Data.

NB! Prøv også

```
1 PRINT"sf1f9OPLØSNING I PRIMFAKTORER"  
  :PRINT:INPUT"TAL";A:PRINT:PRINTA"=";:B=2  
2 IFA/B=INT(A/B)THENA=A/B:PRINTB"*";:IFA  
  > 1THEN2  
3 IFA > 1THENB=B+1:GOTO2  
4 PRINT"vm":RUN
```

LEKTION 17

DEFINERBARE FUNKTIONER

I lektion 15 lavede vi et program, der alfabetiserede ud fra de enkelte karakterers talværdier, ASCII-koden, som du her i bogen kan finde opremset i appendikset. Vi kan også få maskinen til selv at opgive dem. Ordren

```
?ASC("A")
```

betyder således: Opgiv ASCII-koden for A, som er 65. Omvendt betyder

```
?CHR$(65)
```

skriv den karakter, hvis ASCII er 65, altså et A. Funktionen er praktisk i forbindelse med

```
GET
```

Tag programmet fra sidste lektion, hvor vi skulle taste N, Ø, V eller S *fulgt af RETURN*. Her kunne vi i stedet for INPUT have haft GET. Prøv

```
1 INPUT"RETNING(N/Ø/S/V)";A$:PRINTA$
```

og derefter

```
1 PRINT"RETNING(N/Ø/S/V)?"
2 GETA$:IFA$=""THEN2
3 PRINTA$
```

Nu kan vi meget praktisk undvære RETURN, hvad der blandt andet gør programmet lettere tilgængeligt for den »ikke-indviede« bruger. Læg mærke til, at GET ikke »venter« som INPUT. Derfor har vi spærret den inde i en uendelig løkke, der kun brydes, når noget indtastes (når A\$ ikke længere er den tomme streng). Vi kan nu have

```

1 GETA$:IFA$=""THEN1
2 IFA$=CHR$(133)THEN A$=CHR$(144)
3 IFA$=CHR$(137)THEN A$=CHR$(5)
4 IFA$=CHR$(134)THEN A$=CHR$(28)
5 IFA$=CHR$(138)THEN A$=CHR$(159)
6 IFA$=CHR$(135)THEN A$=CHR$(156)
7 IFA$=CHR$(139)THEN A$=CHR$(30)
8 IFA$=CHR$(136)THEN A$=CHR$(31)
9 IFA$=CHR$(140)THEN A$=CHR$(158)
10 PRINTA$;:GOTO1

```

ASCII 133-140 er de 8 FUNKTIONS-TASTER i højre side af tastaturet mærket f1-8. Lige numre får du med SHIFT. De er programmerbare, og du har netop programmeret dem! Vi kunne også have haft

```

1 GETA$:IFA$=""THEN1
2 IFASC(A$)<133ORASC(A$)>140THEN12
3 ONASC(A$)-132GOTO4,6,8,10,5,7,9,11
4 A$=CHR$(144):GOTO12
5 A$=CHR$(5):GOTO12
6 A$=CHR$(28):GOTO12
7 A$=CHR$(159):GOTO12
8 A$=CHR$(156):GOTO12
9 A$=CHR$(30):GOTO12
10 A$=CHR$(31):GOTO12
11 A$=CHR$(158):GOTO12
12 PRINTA$;:GOTO1

```

Udtrykket

ON A GOTO B,C,D,... E

betyder, at værdien af variabelen A bestemmer, om der skal springes til linje B,C,D, osv. Hvis A er 1, vælges den *første* mulighed (det første DESTINATIONS-TAL), hvis 2 den anden, og så fremdeles. De to versioner er selvfølgelig typiske demonstrations-programmer, anskuelige snarere end fikse. Pointen er imidlertid, at maskinen stiller så mange forskellige udtryksmåder til rådighed for brugeren, at det skulle være muligt i hvert enkelt tilfælde at finde den helt rigtige formulering. En anden fiks ting er

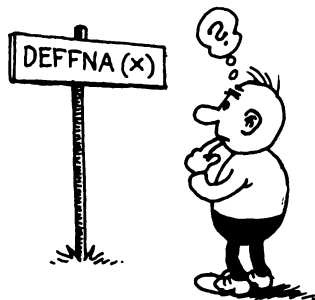
DEFFN

Bogstaverne står for »definer funktion«, og ordren sætter dig faktisk i stand til i begyndelsen af et program at definere dine egne funktioner. F.eks. kan du jo få brug for en, der vælger et tilfældigt tal mellem 1 og et opgivet tal. Du kunne så have

DEFFNA(X)=INT(RND(1)*X)+1

Maskinen protesterer

?ILLEGAL DIRECT
ERROR



= kan ikke bruges uden for programmer, så vi stopper den ind i et:

```
1 DEFFNA(X)=INT(RND(1)*X)+1
2 INPUTB
3 PRINTFNA(B)
```

Her er

A navnet på funktionen (du kunne jo have flere definerede i samme program).

X en PSEUDOVARIABEL, der viser, hvad der sker med det tal, du senere kommer i parentes.

B tallet, det i selve programmet går ud over.

Du kunne også have haft en »fast terning« eller måske to:

```
1 DEFFNA(X)=INT(RND(1)*6)+1:DEFFNB(X)=INT(RND
  (1)*6)+1+INT(RND(1)*6)+1
2 PRINTFNA(X)
3 PRINTFNB(X)
```

Funktion A slår med en terning, B med to. Læg mærke til, at argumentet nu ingenting betyder, men kun er med af hensyn til syntaksen.

NB! Prøv

```
1 A$=A$+CHR$(INT(RND(1)*10)+48):A=A+1:PRIN-
  TA$:FORB=1TO500+A*100:NEXT
2 INPUT"hTAL";B$:IFB$=A$THEN1
3 PRINT"NEJ, "A$:PRINTA*100"POINTS"
```

LEKTION 18

ANDRE FUNKTIONER

I første del af denne bog gennemgik vi begyndelsesgrundene i programmeringssproget BASIC. Vi opdagede, hvad en variabel og et program var, hvordan tastaturet fungerede, samt enkle funktioner og ordrer som GOTO, INPUT, IF, relationer og logiske operationer. I anden del er vi nået et stykke videre, idet vi har stiftet bekendtskab med FOR-NEXT-løkker, strengdeling, subrutiner og sæt. Vore programeksempler har spændt fra simple beregningsprogrammer over tal- og tekstspil til en simulation, en database, alfabetisering og tekstbehandling.

Derimod er vi gået uden om de bekendte TV- og automatspil, 64's lyd- og musikkapacitet, eller kort og godt: Det meste af det, der fik os til at vælge en 64 fremfor en ZX81: Hurtighed, farver og lyd. Når vi har forsømt disse faciliteter så langt, skyldes det, at vi her ikke kan klare os med »almindelig« BASIC, vi må »om bag kulisserne« og erfare lidt om, hvordan maskinen virker, og således skaffe os en kontrol med ikke mindst skærmen, som det blotte og bare kendskab til tastaturet og BASICs syntaks ikke kan give os. Herom BOGENS TREDJE DEL.

I resten af denne del skal vi samle nogle løse ender op. I lektion 19 skal vi lære at oplagre vore programmer på bånd. I lektion 18 skal vi se på nogle forsømte funktioner. Mange af disse er selvfølgelig hovedsagelig til glæde for matematikeren, og dem skal vi gå forholdsvis let hen over. F.eks. er der de såkaldt trigonometriske funktioner:

SIN(X) – SINUS

COS(X) – COSINUS
TAN(X) – TANGENS
ATN(X) – BUETANGENS

LOG(X)

og

EXP(X)

bruges til beregning med naturlige logaritmer. Prøv

?12*34

og derefter

?EXP(LOG(12)+LOG(34))

Noget mere brugbar for almindelige dødelige er

ABS(X)

Prøv

```
1 A=INT(RND(1)*100):B=INT(RND(1)*100):PRINTA"*"B
  ;:INPUTC:IFC=A*BTHEN1
2 PRINT"DU RAMTE"ABS(C-A*B)"FORKERT":GO
  TO1
```

Her er det ikke meningen, at der skal svares ubetinget rigtigt, men at man gør et skøn og derefter ser, hvor nær man kom. Havde vi nu simpelt hen skrevet $C-A*B$, ville vi have fået et negativt tal halvdelen af gangene. ABS smider fortegnet væk.

POS(X)

opgiver, hvor markøren er henne i linjen. Prøv

?TAB(10)POS(0)

Også her er funktionens argument uden betydning.

Vi har allerede været igennem de fleste fejlmeddelelser. Andre, du kan risikere, er

ILLEGAL QUANTITY = for stort argument for den pågældende funktion, altså f.eks. CHR\$(300).

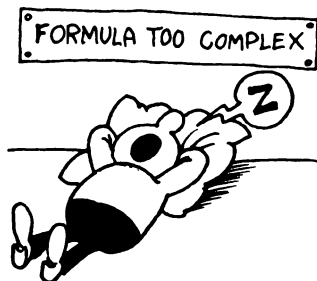
UNDEF'D FUNCTION = du har glemt en DEFFN!

Herudover har maskinen visse begrænsninger. Nogle er selvfølgelig. Man *kan* faktisk ikke dividere med 0, og et forsøg herpå giver

DIVISION BY ZERO

Andre er begrænsninger ved selve maskinen. Det kan (sjældent) ske, at et udtryk bliver for kompliceret for maskinen at arbejde med. Så siger den

FORMULA TOO COMPLEX



Del udtrykket op i to mindre, og det går godt igen. Maskinen kan heller ikke arbejde med strenge på mere end 255 karakterer. Her lyder fejlmeddelelsen

STRING TOO LONG

og vi må ty til sæt.

Måske har du også prøvet at rette i en programlinje og så komme hurtigt ned i bunden med RETURN. Så er du garanteret også af og til endt med en OUT OF DATA. Hvorfor? Jo, da markøren nåede READY, og du derefter tastede RETURN, opfattede maskinen det som den direkte ordre READ Y, og det kan maskinen selvfølgelig ikke, eftersom der ikke er nogen DATA-linjer at læse Y fra! Prøv med SHIFT RETURN i stedet!

Endelig er der faktisk to hele ordrer, vi slet ikke har brugt! Den ene er

REM

der står for REMark, bemærkning, og den betyder faktisk, at maskinen *ikke* skal gøre det, der står i ordren, altså: Dette er ingen ordre til maskinen, men en note til oplysning for den, der læser programmet.

Så har vi

LET

der helt kan undværes.

LETA=27

betyder nemlig ganske det samme som

A=27

```
1 REM LØVEFAMILIER AF FORSKELLIG
  STØRRELSE ("O" < "W" < "Q") OG GAZELLE
  FLOKKE (*)
2 DIMA(25,42),B(25,42):FORA=1TO100 : B (INT(RND(1)
  *23)+2,INT(RND(1)*40)+2)=1:NEXT
3 FORA=1TO100:B(INT(RND(1) *23)+2,INT (RND(1)*
  40)+2)=9:NEXT:PRINT"h"
4 B=B+1:PRINT"hf4f9ANNO"B:A=0 : C = 0 :FORD
  =2TO24:FORE=2TO41:A (D,E) = B (D,E)
5 IFA(D,E)=0THENPRINT"m";
6 IFA(D,E)=1THENPRINT"f2O";:A=A+1
7 IFA(D,E)=2THENPRINT"f2W";:A=A+1
8 IFA(D,E) > 2AND A (D,E) < 9THENPRINT"f2Q";
  :A=A+1
9 IFA(D,E)=9THENPRINT"f1*";:C=C+1
10 NEXT:NEXT:PRINT"f4f9LØVE"A"vmGAZELLE"C"
  vm";
11 FORD=2TO24:FORE=2TO41:IFA (D,E)=0ORA(D,E)=
  9THEN18
12 IFA(D-1,E)=9THENB (D-1,E) = 0 : B (D,E) =A (D,E) +1
13 IFA(D,E-1) = 9THENB (D,E-1) = 0 : B (D,E) = B (D,E) +1
14 IFA(D,E+1) = 9THENB (D,E+1) = 0 : B (D,E) = B(D,E) +1
15 IFA(D+1,E) = 9THENB (D+1,E) = 0 : B (D,E) = B (D,E) +1
16 IFA(D,E) > 3THENB (D,E) =A (D,E) -2: B (INT(RND(1)
  *23)+2,INT(RND(1)*40)+2)=2
17 B(D,E) = B (D,E)-1
18 NEXT:NEXT:FORD=1TOC :E=INT(RND(1) *23)
  +2:F=INT(RND(1) *40)+2
19 IFB(E,F)=0THENB(E,F)=9
20 NEXT:GOTO4
```

LEKTION 19

SAVE

64 er en farve-datamat til cirka tre tusind, og det er selvfølgelig en af dens væsentlige fordele. Samtidig eksisterer der imidlertid en masse tilbehør, hvoraf en del er næsten lige så dyrt, eller endog dyrere end selve maskinen. Det er derfor et nærliggende spørgsmål: Hvor meget behøver jeg?

Et svar er naturligvis: Ikke noget. Alle de i bogen gennemgængede funktioner og programmer kræver kun grundmodellen. En ting er dog mindst talt god at have: Kassettebåndstationen, så vi ikke behøver at taste alle vore programmer ind på ny, hver gang vi tænder for maskinen. Vi står så med en datamat til under 4000 kr., der opfylder alle vore rimelige behov. Selvfølgelig er en printer både sjov og praktisk, den er jo et helt lille trykkeri for sig. Men nødvendig er den ikke. Og 3-5000 ekstra er mange penge. Vi skal her forudsætte, at du har kassettestationen, og lære at gemme vore programmer på bånd. Har du ingen, kan du indtil videre bare springe denne lektion over.



Indtast nu f.eks. LIFE fra lektion 16. Sæt bånd i båndoptageren. Vær sikker på, at det ikke er sikret mod sletning ved, at den lille tap øverst til venstre på båndsiden er trykket ud. Er det sket, kan du gøre det godt igen med et stykke tape. Check ligeledes, at båndet er spolet tilbage. Nu taster du

SAVE”LIFE”

eller blot

SAVE

fulgt af RETURN. I det første tilfælde har du givet programmet et navn, fordi du har i sinde at have flere programmer på et bånd. Lad os blot vælge denne mulighed. Maskinen skriver

PRESS RECORD & PLAY ON TAPE

Du holder REC nede på båndoptageren og trykker PLAY ned, til den låser, og båndet kører. Maskinen siger

OK

SAVING LIFE

og efter et stykke tid

READY.

Læg mærke til, at båndet er standset. Tryk nu på STOP og derefter REW. STOP igen, når båndet er spolet tilbage. Du har nu formodentlig en optagelse, men vil du være sikker, kan du VERIFICERE det. Tast

VERIFY

fulgt af RETURN. Maskinen forlanger

PRESS PLAY ON TAPE

og kvitterer med OK, når du trykker PLAY, endvidere

SEARCHING

dvs. »jeg søger«, nemlig efter et program at verificere. Da vi ikke har givet besked om, hvilket program den skal lede efter, tager den blot det første på båndet. Efter en tid får vi

FOUND LIFE

så

VERIFYING

og endelig

OK

READY.

Det eneste, der kan gå galt, er sådan set en fejl i båndet, der giver VERIFY ERROR. Så må vi bare prøve igen med et andet bånd. Spol tilbage og tryk på STOP, og så en gang til, denne gang for EJECT. Låget springer op, og vi kan tage vores bånd, anbringe det et sted uden for fjernsynets umiddelbare nærhed (et TV er en elektromagnetisk dingenot og kan slette båndet) og slukke for datamaten. Nu leger vi, at det er blevet torsdag, og tænder igen. Vores program er naturligvis pist væk. Læg nu båndet i båndoptageren og tast

LOAD

Der er jo kun det ene program på båndet. Igen får vi at vide, at vi skal trykke på PLAY, vi gør det, maskinen svarer OK og går i gang med at lede. Vi får

FOUND LIFE

og derefter

LOADING

samt et READY, når den er færdig. Tast nu LIST og RUN, og vi har vores program som før. Havde vi nu ikke verificeret, og båndet havde været defekt, havde vi her fået LOAD ERROR. Så havde det bare været for sent . . .

NB! Prøv

```
1 A$="1V2":?"f9TIPSKUPONS":FORA=1TO13:B=INT  
(RND(1)*3)+1:?"TAB(B) MID$(A$,B,1):NEXT
```

LEKTION 20

SPIEL

```
1 PRINT"hDU FØRER RUMSKIBET CO
  MMODORE":A=1000
2 B=1000+RND(1)*1000:C=0:D=0:PRINT"FJENDTLI
  GT RUMSKIB I SIGTE"
3 PRINT"f9POINTS"E:PRINT"sAFSTAND"B:PRIN
  T"RELATIV HASTIGHED"F:PRINT"ENERGI"A
4 PRINT"KRAFTSKJOLD"G:PRINT"SKADE I SKIB"
  NT(H):PRINT"SKADE I ANDET SKIB"INT(C)
5 IFRND(1)<.9THENPRINT"s9FJENDEN SKYDER":
  H=H+RND(1)*(1000-B-G)/10
6 PRINT"sF1 HASTIGHED":PRINT"F3 KRAFTSKJ
  OLD":PRINT"F5 LASER":PRINT"F7 FASER"
7 PRINT"s9ORDRE?"
8 GETA$:IFA$=""THEN8
9 I=ASC(A$)-132:IFI=1THENINPUT"HASTIGHED
  (0-100)":F
10 IFI=2THENINPUT"KRAFTSKJOLD (0-100)":G
11 IFI=3THENA=A-10:C=C+RND(1)*(1000-B)/10
12 IFI=4THENA=A-20:C=C+RND(1)*(1000-B)/5
13 A=A-(F+G)/10:B=B-F-RND(1)*100:IFC < 0THENC
  =0
14 IFH < JTHENH=J
15 IFC >=100THENPRINT"hf9FJENDEN ØDELAGT":
  E=E+(25-D)*100:J=H:PRINT"NYT":GOTO2
16 IFA <=0THENPRINT"f9ENERGI 0":GOTO20
17 IFB <=0THENPRINT"f9SAMMENSTØD":GOTO20
18 IFH >=100THENPRINT"f9COMMODORE ØDELA
  GT":GOTO20
```

```

19 D=D+1:PRINT"h":GOTO3
20 PRINT"f9POINTS"E

```

Ovenstående program indeholder en stor del af de ordrer og funktioner, vi har gennemgået i denne bogs to første dele. Gennemgå og forstå det. Prøv at lave det om. Som en hjælp er her en liste over variabler:

```

A ENERGI
B AFSTAND
C SKADE I ANDET SKIB
D OMGANG
E POINTS
F RELATIV HASTIGHED
G KRAFTSKJOLD
H SKADE I SKIB
I ORDRE
J SIDSTE SKADE I SKIB

```

Prøv på samme måde

```

1 FORA=1TO6:A(A)=INT(RND(1)*10)+1:NEXT:PRINT
  "FIND PRINSESSEN"
2 PRINT"f1RETNING(N/S/V/Ø)?"
3 GETA$:IFA$=""THEN3
4 IFA$="N"THENA(1)=A(1)-1
5 IFA$="S"THENA(1)=A(1)+1
6 IFA$="V"THENA(2)=A(2)-1
7 IFA$="Ø"THENA(2)=A(2)+1
8 IFA(1)=A(3)AND A(2)=A(4)THENPRINT"f5f9SPIST!":E
  ND
9 IFA(1)=A(5)AND A(2)=A(6)THENPRINT"f5f9ELSKED
  E!":PRINT"POINTS"(50-E)*10:END
10 IFRND(1)<.75THEN15

```



```

11 PRINT"f2UHYRET ER I ";:IFA(3) < A(1)THEN A
    (3)=A(3)+1:PRINT"NORD"
12 IFA(3) > A(1)THEN A(3)=A(3)-1:PRINT"SYD"
13 IFA(4) < A(2)THEN A(4)=A(4)+1:PRINT"VEST"
14 IFA(4) > A(2)THEN A(4)=A(4)-1:PRINT"ØST"
15 IFRND(1) < .75 THEN 20
16 PRINT"f6PRINSESEN ER I ";:IFA(5) < A(1)THEN A
    (5)=A(5)-1:PRINT"NORD"
17 IFA(5) > A(1)THEN A(5)=A(5)+1:PRINT"SYD"
18 IFA(6) < A(2)THEN A(6)=A(6)-1:PRINT"VEST"
19 IFA(6) > A(2)THEN A(6)=A(6)+1:PRINT"ØST"
20 FOR A=1 TO 6: IFA(A)=0 THEN A(A)=10
21 IFA(A)=11 THEN A(A)=1
22 NEXT A: A=ABS(A(1)-A(3)): B=ABS(A(2)-A(4)): C=ABS
    (A(1)-A(5)): D=ABS(A(2)-A(6))
23 IFA < 2 AND B < 2 THEN PRINT"f2f9MØRK SKYGGE"
24 IFA < 4 AND B < 4 THEN PRINT"f2f9TUNGE SKRID
    T"
25 IFC < 2 AND D < 2 THEN PRINT"f6f9SKRIG"
26 IFC < 4 AND D < 4 THEN PRINT"f6f9LØBENDE FØD
    DER"
27 PRINT"f8CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    CCCCCCCCCCCC": E=E+1: GOTO 2

```

Så enkle disse spil end er, fortæller de dog noget om »mekanikken« i et program. Prøv programmet og besvar følgende spørgsmål:

1. Hvordan ved maskinen, hvem der er hvor i »labyrinten«?
2. Hvorfor går uhyret og prinsessen langsommere end du?
3. Hvordan kan maskinen fortælle dig, om f.eks. uhyret er nord eller syd for dig?
4. Det hele foregår egentlig på overfladen af en kugle. Hvordan det?

5. Hvordan kan maskinen fortælle dig, at f.eks. prinsessen er nær ved dig?
6. Uhyret jager dig, prinsessen løber væk fra dig. Hvordan ordner maskinen det?
7. Somme tider er prinsessen skiftevis f.eks. nord og syd for dig. Hvorfor? Hvordan kan du så finde hende?

Find frem, hvad du synes, er svagheder ved dette spil, og lav det om, til det passer dig. Lav andre spil. Hvis du nøjes med at læse de første 20 lektioner igennem, lærer du intet. *Alle principperne skal bruges af dig i dine egne programmer!* Det er den gyldne regel for hele denne bog, og hvis du ikke overholder den, kan du lige så godt lægge bogen fra dig med det samme. Du lærer alligevel ikke at svømme uden at blive våd.

Hvis du derimod nøje har læst de første 20 lektioners gennemgang og beskrivelse af funktionerne og brugt hver eneste af dem i det mindste et par gange, skulle du faktisk allerede nu være i besiddelse af pæne forudsætninger i BASIC. Du har dog endnu langt fra udnyttet din maskines muligheder fuldt ud. At du nu kender til de store tandhjul, betyder ikke, at du har fuld kontrol over maskinen. Der er trods alt langt fra tal på en skærm til et fungerende TV-spil. Denne vej vil vi imidlertid tilbagelægge i tredje del af bogen.

Så se lige engang tilbage; har du nu husket alt (eller i al fald det meste)? Godt, så fanger vi an.

NB! Prøv

```

1 FORA=1TO6:READA$(A):A(A)=100:NEXT:A=1000
2 PRINT"hf9ANTALmAKTIEmKURS":PRINT:FORB
  =1TO6:PRINT"f9" B(B) B ;A$(B) A(B):NEXT
3 PRINT:PRINT"KAPITAL:"A:PRINT:INPUT
  "KØB/SALG (K/S) " ;A$ :IFA$="K"THEN6
4 IFA$="S"THEN8

```

```

5 GOTO10
6 GOSUB14:IFA(B)*C > A THEN6
7 B(B)=B(B)+C:A=A-A(B)*C:GOTO10
8 GOSUB14:IFC > B(B) THEN8
9 B(B)=B(B)-C:A=A+A(B)*C
10 FORB=1TO6:A(B)=A(B)+INT(RND(1)*21)-10:IFA(B)
    < 10 THENA(B)=10
11 NEXT:A=A-10:IFA < 0 THENPRINT:PRINT"DU ER
    FALLIT":END
12 IFA > 10000 THENPRINT:PRINT"DU ER BØRSMA
    TADOR":END
13 GOTO2
14 PRINT:INPUT"AKTIENUMMER";A$:IFASC(A$)
    < 49 ORASC(A$) > 54 THEN14
15 B=VAL(A$):PRINT:INPUT"HVOR MANGE";C:RE
    TURN
16 DATAJENSEN,OLSEN,KLAUSEN,FEDESEN,TOM
    SEN,BORGEN

```


TREDJE DEL
Skærm og karakterer

LEKTION 21

BITS OG BYTES

```
1 A$="nsøv"  
2 PRINTMID$(A$,INT(RND(1)*4)+1,1)"f1X";:FORA=1T  
O100:NEXT:PRINT"vmv";:GOTO2
```

Som du vil se, er der kommet en flue ind i maskinen.

Dette undgås bedst ved altid at holde vinduerne lukkede, mens man arbejder med den.

En vis primitiv illusion af bevægelse er altså mulig uden andre forkundskaber end dem, vi er i besiddelse af på nuværende tidspunkt. Skal vi videre, må vi imidlertid vide lidt mere om, hvordan maskinen *fungerer*.

I skolen lærte vi at regne med tal på mere end et ciffer ved f.eks. at opsplitte 1234 i

1 tusind
2 hundrede
3 tiere
4 enere

Vores »almindelige« talsystem er et titalssystem, dvs. det består af 10 tal: 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9. Når vi har talt op til 9, får vi 0 igen på ener-pladsen og begynder at tælle på tier-pladsen med 1, op til 9 tiere og 9 enere, hvor vi må tage hundrede-pladsen i brug.

Vi kan imidlertid også lave et talsystem med kun to tal, 0 og 1. Nu skal vi skifte plads allerede, når der står 1 på denne plads, det højest tilladelige. Vi skifter altså første gang plads

ved 2, der bliver 10. Derefter bliver 3 til 11 og 4 bliver til 100. Vi har altså

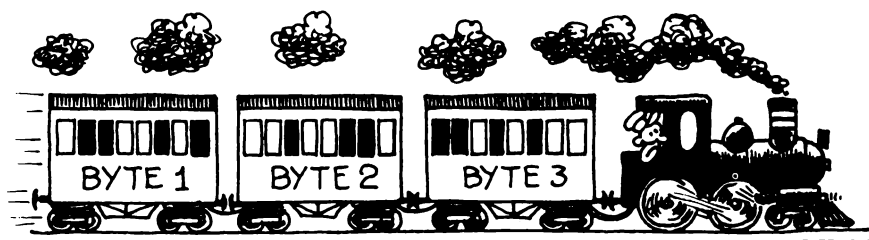
TOTALSSYSTEM	TITALSSYSTEM
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16

Upraktisk? Ikke for en maskine. Nu kan nemlig ethvert tal udtrykkes ved en serie elementer, der enten er tændte eller slukkede. Tallet 13 kræver fire lamper, og 1101 udtrykkes

TÆNDT TÆNDT SLUKKET TÆNDT

altså tændt for 1 og slukket for 0. Totalssystemet hedder på engelsk BINARY SYSTEM, og derfor kalder man et element, der ved at være tændt eller slukket udtrykker et ciffer i totalssystemet, BINARY DIGIT (digit = ciffer) eller simpelt hen BIT. For at få »rigtige« tal, er disse bits ordnet i enheder på 8 bits, og disse enheder kaldes en BYTE. En byte kan (regn efter) udtrykke et tal mellem 0 og 255. Nu ved du,

hvad det vil sige, at du, når du tænder for maskinen, har 3583 bytes til rådighed, og hvorfor en byte cirka kunne oversættes med en karakter. Der er 256 ASCII-koder, hvilket vil sige, at en hvilken som helst af ASCII-systemets karakterer kan udtrykkes ved et tal mellem 0 og 255 og altså af 1 byte. Læg mærke til, at det ikke er ligegyldigt, hvilke bits der er tændt i en byte. 10110010 er noget andet end 11001001. På samme måde skal flere bytes naturligvis også arbejde sammen, når hele BASIC-sprogets utallige muligheder skal udtrykkes. Hver byte har sin plads eller ADRESSE. MASKINE er jo heller ikke det samme som ANEMISK.



Bytes tælles i K. 1 K eller kilo-byte er 1024 bytes. En 16K hukommelsesudvidelse vil altså give dig godt og vel 16000 bytes ekstra at arbejde med. Alt dette er RAM eller Random Access Memory, dvs. hukommelse, som du kan bruge, som du vil, til programmer, variabler, etc. Men maskinen indeholder naturligvis også en for-programmeret viden om, hvordan man lægger sammen, uddrager kvadratrødder, eller blot oversætter dit halv-engelske BASIC til »binærkode« eller »maskinkode«. Denne form for hukommelse kaldes ROM eller Read Only Memory. 64 har 26K ROM, så det er ikke så lidt, den ved, endnu før du har fortalt den noget.

RAM-adresserne er opdelt i større »byer«, hver med deres egen funktion. Der er særlige områder for oplagring af

programmer, variabler, skærbilledet, etc. Herudover er der de såkaldte SYSTEMVARIABLER, der fortæller maskinen noget om, i hvilken »tilstand« den er. Den »egentlige« maskine er nemlig den såkaldte CPU, »Central Processing Unit«. Det er »regneværket« i 64, og det står i forbindelse med alle de forskellige former for hukommelse.

SYSTEMVARIABLERNE skal vi se på i næste lektion. Her skal vi skrive et lille nemt program, der kan »oversætte« et tal i totalssystemet til et tal i titalssystemet inden for intervallet 0-255, noget, vi snart skal få brug for:

```
1 INPUT A$:FOR A=1 TO 8:B=B+VAL(MID$(A$,A,1))*2p(8-A):NEXT:PRINT B
```

Husk, når du afprøver programmet, at sætte nuller forrest, sådan at der i alt er 8 cifre i inputtallet.

Vi kan naturligvis også gå den anden vej:

```
1 INPUT A:FOR B=7 TO 0 STEP -1:IF A >= 2pB THEN PRINT"1";:A=A-2pB:GOTO 3
2 PRINT"0";
3 NEXT
```

Prøv nogle tal igennem på de to programmer. Prøv at finde nogle regneregler for totalssystemet. Du skal føle dig forholdsvis sikker i disse tal, før du går videre . . .

NB! Prøv

```
1 REM FIDUS: OPTEGN OASERNES PLACERING I
  HVERT SPIL OG RID TILBAGE EFTER VAND
2 FOR A=1 TO 10:READ A$(A):A(A)=INT(RND(1)*1000):NEXT:A=1:B=10:C=10:D=10:E=1
3 PRINT"hf4"A". DAG":PRINT:PRINT"f1DU HAR
  REJST"F"KM":PRINT"DU HAR"B"L VAND"
```

```

4 PRINT"f6":IFC > 0THENPRINT"DU ER TIL
  HEST":PRINT"HESTEN ERm"A$(C)
5 IFC=0THENPRINT"DU ER TIL FODS"
6 PRINT:PRINT"f2DU ERm"A$(D):IFA > 10ANDG
  < 100THENPRINT"f9RØVERE"G"vmKM BORTE"
7 FORH=1TO10:IFABS(A(H)-F) < 10ANDC >
  0THENPRINT"f9HESTEN LUGTER VAND"
8 IFABS(A(H)-F) < 5THENPRINT"f9OASE":B=10:D
  =10:IFC > 0THENC=10
9 NEXT:1FRND(1) < .01THENPRINT"f9SAND
  STORM":B=B-5:C=C-5:D=D-5
10 PRINT:PRINT"f5VIL DU"
11 PRINT"1. REJSE HURTIGERE":PRINT"2. REJSE
  LANGSOMMERE":PRINT"3. HVILE MEGET"
12 PRINT"4. DRIKKE":PRINT"5. VENDE OM"
  :PRINT"6. REJSE SOM FØR"
13 GETA$:IFA$=""THEN13
14 IFASC(A$)< 49ORASC(A$)> 54THEN13
15 ONVAL(A$)GOTO16,19,23,25,30,32
16 H=H+INT(RND(1)*10):IFC > 0THENC=C-3
17 IFC=0THEND=D-2
18 D=D-1:GOTO34
19 H=H-INT(RND(1)*10):IFH < 1THENH=1
20 IFC > 0THENC=C-2
21 IFC=0THEND=D-1
22 D=D-1:GOTO34
23 H=0:IFC > 0THENC=C-1
24 D=D-1:GOTO34
25 IFB > 0THEN28
26 IFC > 0THENC=C-1
27 D=D-1:GOTO34
28 B=B-1:IFC > 0THENC=C-1
29 D=D+1:GOTO34
30 E=E*-1:IFC > 0THENC=C-1
31 D=D-1:GOTO34

```

```

32 IFC > 0THENC=C-1
33 D=D-1
34 A=A+1:F=F+(C + D +H) *E:IFF < 0THENPRINT"f9
    DU ER HJEMME IGEN":END
35 IFA < 11THEN39
36 IFF > ITHENI=I+INT(RND(1) *10)
37 IFF < ITHENI=I-INT(RND(1) *10)
38 G=ABS (F-I):IFG < 5THENPRINT"f9RØVERNE
    INDHENTEDE DIG":END
39 IFC > 10THENC=10
40 IFC < 0THENC=0
41 IFD > 10THEND=10
42 IFD < =0THENPRINT"f9DU ER DØD":END
43 GOTO3
44 DATADØENDE,MEGET SYG,SYG,UTILPAS, ME
    GET VARM, TØRSTIG, VARM, VELTILPAS,RASK,I
    FORM

```

LEKTION 22

SYSTEMVARIABLER

Skift farve til sort. Maskinen vil skrive sort, indtil du forlanger en anden farve. Men hvordan kan den nu vide det? Du har jo for længe sluppet farvetasten. Et eller andet sted i maskinen må den være »slået til« rød farve.

Faktisk skynder maskinen sig ned i ADRESSE 646 og ser efter, hvad der står der, hver gang den skal til at skrive en karakter. Hver farve har nemlig sin FARVEKODE.

- 0 SORT
- 1 HVID
- 2 RØD
- 3 CYAN
- 4 VIOLET
- 5 GRØN
- 6 BLÅ
- 7 GUL
- 8 ORANGE
- 9 BRUN
- 10 LYSERØD
- 11 GRÅ 1
- 12 GRÅ 2
- 13 LYSEGRØN
- 14 LYSEBLÅ
- 15 GRÅ 3

Det får vi brug for i anden sammenhæng. Forklaringen er nu simpel: Så længe værdien i adresse 646 er 0, så længe vil maskinen skrive sort.

Det er selvfølgelig noget af en påstand, men vi kan nu også bevise den. Vi har nemlig en funktion, der kan kigge direkte ind i en adresse; faktisk betyder

PEEK

på engelsk »kigge«. Prøv nu

PRINTPEEK(646)

2, ikke sandt? Skift farve og prøv igen. Prøv nu

POKE646,5

Maskinen skriver *READY med grønt*. Ordren POKE virker nemlig den modsatte vej: Den lægger en værdi direkte ind i en adresse. Her skiftede vi 2 ud med 5, og maskinen skriver grønt!

Prøv at slå tilbage i første dels lektion 4. Tag eksemplet

```
10 PRINT"f664"
```

```
20 PRINT"FRA f2COMMODORE"
```

```
30 PRINT"f1DEN PERSONLIGE DATAMAT"
```

Vi kunne sådan set også have haft

```
10 POKE646,5:PRINT"VIC 20"
```

```
20 PRINT"FRA ":POKE646,1:PRINT"COMMODORE"
```

```
30 POKE646,0:PRINT"DEN PERSONLIGE DATAMAT"
```

Ikke særlig praktisk? Nej, vel? Og her er sådan set sagens kerne. I langt de fleste tilfælde er det langt bedre at forlade sig på »almindelig« BASIC og styre maskinen fra tastaturet. Somme tider er det det imidlertid ikke. Til andre kan man slet ikke opnå den virkning, man tilstræber, uden POKE.



646 er en SYSTEMVARIABLE. Vi kan sådan set ikke hente noget ud af den med POKE, som vi ikke kan få meget enklere fra tastaturet. Men tag så ADRESSE 53280. Prøv

POKE53280,6

Skærmkanten, som vi er blevet så vant til at se, forsvandt. Det vil sige, det gjorde den sådan set ikke. Den fik bare samme farve som tekstfeltet, blå. Prøv

POKE53280,2

Rød kant! Faktisk kan vi vælge mellem ikke mindre end 16 kantfarver og 16 skærmfarver.

Den sidste står i 53281.

```
1 PRINT"SKRIV ELLER LØS KODE":INPUT"(S/L)"
  ;A$:IFA$="S"THEN4
2 IFA$="L"THEN9
3 RUN
4 INPUT"KLARTEKST";A$:INPUT"KODENUMMER
  (1-25)";A
```

```

5  FORB=1 TO LEN(A$) : C=ASC (MID$ (A$,B,1)) : IFC
   =32 THEN PRINT " " ;: GOTO 8
6  C=C+A: IFC > 90 THEN C=C-26
7  PRINT CHR$ (C);
8  NEXT: PRINT: RUN
9  INPUT "KODE TEKST" ; A$ : INPUT "KODE NUMMER
   (1-25) " ; A
10 FORB=1 TO LEN(A$) : C=ASC (MID$ (A$,B,1)) : IFC
   =32 THEN PRINT " " ;: GOTO 13
11 C=C-A: IFC < 65 THEN C=C+26
12 PRINT CHR$ (C);
13 NEXT: PRINT: RUN

```


LEKTION 23

SKÆRMEN

Prøv

```
1 PRINT"h"
2 POKE53280,0:POKE53281,0
3 POKE1500,1:GOTO3
```

De første to linjer indeholder ingen hemmeligheder for os. Skærmen slettes og farves sort. Men 3? Et hvidt A kommer til syne. Vi har ikke tastet noget A. Vi har POKEt det direkte ind i den adresse, der oplagrer, hvad der står på det pågældende felt på skærmen. Prøv nu (efter RUN STOP og RUN STOP/RESTORE)

```
1 PRINT"h"
2 POKE53280,0:POKE53281,0
3 FORA=1024TO2023:POKEA,1:NEXT:GOTO3
```

Nemlig, hvad der står på skærmen, oplagres i adresserne 1024–2023 efter formlen

`LINJE*40+PLADS+983`

Hvad er det så, der står i adresserne? Ja, ASCII er det åbenbart ikke, for ASC(1) er ikke A. Det er de såkaldte SKÆRM-KODER, som du kan finde i denne bogs appendiks.

Men hvor står det, at Aet skal være hvidt? Ingen steder, og det er det sådan set heller ikke, det er farveløst. Skal vi

have farve på, må vi poke den ind i en tilsvarende del af maskinen, der oplagrer hvert enkelt skærmfelts farve. Den strækker sig over adresserne 55296-56295 og går efter formlen

`LINJE*40+PLADS+55255`

Farvekoderne har vi fra lektion 2 og siger

```
1 PRINT"h"
2 POKE53280,1:POKE53281,1
3 POKE10*40+10+983,83
4 POKE10*40+10+55255,2:GOTO4
```

eller

```
1 Slet skærm
2 Helt hvid skærm
3 Hjerter på 10. linje 10. plads
4 Karakteren på 10. linje 10. plads rød
```

Vi skulle jo imidlertid også gerne have lidt bevægelse i tingene. Prøv

```
1 PRINT"h"
2 POKE53280,1:POKE53281,1
3 FORA=1TO40
4 POKE10*40+A+983,81
5 POKE10*40+A+55255,2
6 FORB=1TO100:NEXT
7 POKE10*40+A+983,32
8 NEXT
9 GOTO3
```

eller

- 1 Slet skærm
- 2 Helt hvid skærm
- 3 FOR
- 4 Bold på 10. linje A. plads
- 5 Karakteren på 10. linje A. plads rød
- 6 Vent 1/10 sekund
- 7 Mellemrum på 10. linje A. plads
- 8 NEXT
- 9 Om igen

Når vi har set på bolden i 1/10 sekund, trykker vi et mellemrum oven på den, altså sletter den, og trykker den så igen en plads længere til højre. Illusionen er bevægelse!

Når bolden har nået højrekant, begynder den forfra i venstre side. Men kunne det nu ikke være morsommere, hvis den »stødte mod« højrekanten og løb den anden vej? Prøv

```

1 PRINT"h"
2 POKE53280,1:POKE53281,1
3 A=1
4 B=1
5 POKE10*40+A+983,81
6 POKE10*40+A+55255,2
7 FORC=1TO100:NEXT
8 POKE10*40+A+983,32
9 A=A+B
10 IFA=1ORA=40THENB=-B
11 GOTO5

```

Vi har her udskiftet kontrolvariablen A med en almindelig variabel, der bliver B større for hver omgang gennem løkken. I første omgang er B 1, og A gennemløber derfor værdierne 2, 3, 4 osv. Men når A bliver 40 (når højrekant), bliver B til -B, altså 1 til -1, og A antager værdierne 39, 38, 37

etc. Til sidst er A 1, på hvilket tidspunkt -B bliver -(-B), altså 1 igen, og bolden atter drager mod højre. Når du er sikker på, at du har forstået dette til bunds, kan du gå videre med

```
1 PRINT"h":POKE53280,6:POKE53281,1:A=1:B=1:C=1:
  D=1
2 POKEA*40+B+983,81:POKEA*40+B+55255,2:POKEA
  *40+B+983,32:A=A+C:B=B+D
3 IFA=1ORA=25THENC=-C
4 IFB=1ORB=40THEND=-D
5 GOTO2
```

eller

- 1 Gør skærmen parat og initialiser
- 2 Tegn og mal bold, vent, slet og flyt bold
- 3 Skift flag for A
- 4 Skift flag for B
- 5 Om igen

Læg mærke til, at vi kalder C og D for FLAG. Flag er vore egne private systemvariabler. Når flag 2 er hejst på stang 646, ved maskinen, at den skal skrive med rødt. På samme måde fortæller vi nu maskinen, at når flag D er hejst (1), skal bolden mod højre, når det er strøget (0), mod venstre.

Vi har nu lært lidt om, hvordan vi får ting til at bevæge sig på skærmen. Eksperimentér selv videre. I næste lektion skal vi prøve, om vi ikke kan få et lille spil ud af vores dansende bold. Døren er åbnet på klem ind til TV-spillene. Lad os se, om vi ikke kan få den helt op.

LEKTION 24

BEVÆGELSE

```
1 POKE53280,0:POKE53281,1:PRINT"hf1
  NIVEAU(1-9)?"
2 GETA$:IFA$=""THEN2
3 IFASC(A$) < 49ORASC(A$) > 57THEN2
4 A=983:B=55255:C=9:D=11:E=1:F=1:G= VAL(A$):P
  OKE650,128
5 PRINT"h":FORH=2TO7:FORI=1TO40:POKEA+H*40
  +I,250:POKEB+H*40+I,H:NEXT:NEXT
6 IFC=0THENPOKE650,0:GOTO6
7 H=8:I=2:IFRND(1) < .5THENI=3
8 GETA$:POKEA+H*40+I,32:POKEA+D+1000,32:H=H
  +E:I=I+F
9 IFPEEK(A+H*40+I)=250THENJ=J+(8-H) *G:GOSUB
  19:E=-E
10 IFH=24ANDD < > ITHENC=C-1:GOSUB19:GO
  TO6
11 IFA$="Z"THEND=D-1
12 IFA$="M"THEND=D+1
13 IFD=0THEND=1
14 IFD=41THEND=40
15 POKEA+H*40+I,81:POKEB+H*40+I,0:POKEA+D+10
  00,120:POKEB+D+1000,0
16 IFH=1ORH=24THENE=-E
17 IFI=1ORI=40THENF=-F
18 FORK=1TO(9-G)*10:NEXT:GOTO8
19 PRINT"hf1f9LIV"C"POINTS":RETURN
```

- A 983
- B 55255
- C LIV
- D BAT
- E BOLDENS LINJEFLAG
- F BOLDENS PLADSFLAG
- G NIVEAU
- H BOLDENS LINJE
- I BOLDENS PLADS
- J POINTS
- K PAUSE

1. Hvid skærm med sort kant. Niveau?
2. Tag imod niveau.
3. Korrekt indtastning?
4. Initialisering. Hvad gør POKEn? Prøv at udelade den og se, hvad der sker!
5. Lav mursten.
6. Hvis ikke flere liv, tilbage til normaltilstand. Uendelig løkke, billedet fryses (du skal bruge RUN STOP/RE-STORE for at komme ud).
7. Boldens startposition.
8. Check for indtastning. Slet gammel bold, gammelt bat. Position for ny bold.
9. Hvis mursten ramt, pointsforøgelse, skift boldens linje-flag.
10. Rammer bat ikke bold, – 1 liv, ny omgang.
11. Bat til venstre?
12. Bat til højre?
13. Bat for langt til venstre?
14. Bat for langt til højre?
15. Tegn og mal bold og bat.
16. Hvis bold rammer top eller bund, skift linje-flag.
17. Hvis bold rammer side, skift pladsflag.
18. Pause svarende til niveau.

19. Skriv liy tilbage, points.

Som overalt i bogen er programmerne det vigtigste. Det er ved at analysere og forbedre dem, du lærer at programmere. Vær helt sikker på, at du har forstået til bunds, hvordan ovenstående fungerer.

Lav derefter programmer, hvor

1. Farverne er andre.
2. Spillet er lettere, fordi farten på bolden er mindre.
3. Spillet er lettere, fordi du ikke behøver at ramme så nøjagtigt. Prøv en formel af typen

IFABS(D-I) < 2 THEN

altså hvis bare ikke der rammes mere end én plads forkert; så . . .

4. Bolden ikke skifter linjeflag, hver gang den rammer en mursten.

```
1 POKE650,128:POKE53280,5:POKE53281,5:TI$="000
  000":A=9:B=11:C=50:PRINT"h"
2 PRINTTAB(A)"f1c+f4f9mmmmf1c+":IFRND(1)<.1T
  HENPRINTTAB(A+INT(RND(1)*5)+1)"nf3p"
3 IFD > 22ANDPEEK(1023+B) < > 160THEN3
4 POKEB+1023,90:POKEB+55295,2:IFRND(1) < .5THE
  NA=A-1
5 IFRND(1) < .5THENA=A+1
6 IFA=0THENA=1
7 IFA=34THENA=33
8 GETA$:IFA$="Z"THENB=B-1
9 IFA$="M"THENB=B+1
10 IFA$="1"THENC=50
11 IFA$="2"THENC=25
12 IFA$="3"THENC=0
```

```

13 D=D+1:IFD=1000THENPRINTTI$:END
14 FORE=1TOC:NEXT:GOTO2

```

- A VEJ
- B BIL
- C GEAR
- D AFSTAND
- F PAUSE

1. Grøn skærm, start ur, initialiser, slet skærm.
2. Vej, modgående færdsel.
3. Ulykke?
4. Bil. Vej til venstre?
5. Vej til højre?
6. Vej for langt til venstre?
7. Vej for langt til højre?
8. Bil til venstre?
9. Bil til højre?
10. 1. gear?
11. 2. gear?
12. 3. gear?
13. Hvis strækning tilbagelagt, angiv tid.
14. Pause svarende til gear.

Prøv andre variationer.

NB! Prøv også

```

1 PRINT"h":POKE53280,0:53281,0:A=9:B=9:C=200
2 POKEA*40+B+983,32:IFRND(1)<.5THENA=A-1
3 IFRND(1)<.5THENA=A+1
4 IFRND(1)<.5THENB=B-1
5 IFRND(1)<.5THENB=B+1
6 IFA=7THENA=8
7 IFA=13THENA=12

```



```

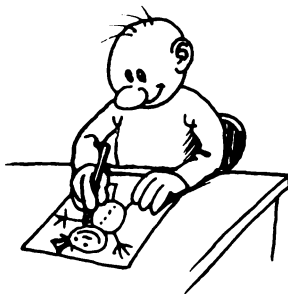
8 IFB=7THENB=8
9 IFB=13THENB=12
10 POKEA*40+ B +983,87:POKE1393,43 :GETA$:IFA$
    < > ""THEN13
11 IFA=10ANDB=10THEN11
12 C=C-10
13 C=C-1:IFC < 0THENC=0
14 PRINT""hf2"C"vm":IFC = 0THEN14
15 GOTO2

```

LEKTION 25

BETINGEDE UDTRYK

```
1 POKE650,128:POKE53280,1:POKE53281,1:A=10:B=1
  0:PRINT"h"
2 POKEA*40+B+983,160:POKEA*40+B+55255,C:GETA$:
  IFA$="W"THENA=A-1
3 IFA$="Z"THENA=A+1
4 IFA$="A"THENB=B-1
5 IFA$="S"THENB=B+1
6 IFA$="F"THENGOSUB12
7 IFA=0THENA=1
8 IFA=26THENA=25
9 IFB=0THENB=1
10 IFB=41THENB=40
11 GOTO2
12 GETA$:IFA$=""THEN12
13 IFASC(A$)<49ORASC(A$)>56THEN12
14 C=VAL(A$)-1:RETURN
```



A LINJE
B PLADS
C FARVE

1. Hvid skærm, prikkens startposition, slet skærm.
2. Tegn og mal prik, check for indtastning, prik op?
3. Prik ned?
4. Prik til venstre?
5. Prik til højre?
6. Farveskift?
7. Prik for langt oppe?
8. Prik for langt nede?
9. Prik for langt til venstre?
10. Prik for langt til højre?
11. Om igen.
12. Check for indtastning af farve.
13. Korrekt indtastning?
14. Farveskift.

Dette program er egentlig på 23 linjer. Linje 1 er således f.eks. sammensat af 5 linjer. Denne sammenpresning foretager vi givetvis af hensyn til den anvendte hukommelse, men dog i langt højere grad på grund af den større overskuelighed. Vi må kunne overse et program for at gøre det så godt som muligt. Idealet bliver det kompakte, hurtige, effektive, overskuelige og letforståelige program. Jo mere vi kan presse linjer, der hjælper hinanden med at udføre bestemte opgaver, sammen til faste, omflyttelige »klodser«, jo mere kan vi koncentrere os om de store linjer i vores program. Det synes måske krukke i programmer af denne størrelse, men forestil dig et program af tidobbelt længde. Så er det rart, hvis vi kan finde initialisationen og ikke løber vild i et virvar af overflødige GOTOer og linjenumre.

Betragter vi nu vores program med disse øjne, må det slå os, at der er noget påfaldende »uforkortet« ved linjerne 3-10. Men hvad skal man gøre, det er IF-linjer, som man ikke uden videre kan hægte sammen. Her kommer imidlertid et andet fænomen os til hjælp: SANDHEDSVÆRDIEN og DET BETINGEDE UDTRYK.

Hvad betyder egentlig

IFA=1THEN666

Jo, maskinen skal gå til linje 666, hvis og kun hvis det er opfyldt eller SANDT, at $A=1$. Men hvordan ved maskinen nu det? Jo, den tilkender nemlig helt automatisk enhver relation et tal, afhængigt af, om relationen er sand eller falsk, en såkaldt SANDHEDSVÆRDI, nemlig

-1 (minus 1) hvis relationen er sand

og

0 (nul) hvis relationen er falsk

Det kan du selv kontrollere. Skriv

? "2+2" (skriv strengen "2+2")

derefter

? 2+2 (skriv resultatet af 2+2)

og endelig

? 2+2=4 (skriv SANDHEDSVÆRDIEN af relationen $2+2=4$)

Er du med? Prøv nu

? 2+2=5

Eller

?2+2=4AND2+2=5

Forklar det sidste resultat. Lad os nu vende tilbage til vores program. Tag f.eks.

IFA\$="Z"THEN A=A+1

Hvad, om vi i stedet skrev

A=A-(A\$="Z")

Ja, hvis A\$="Z", så er sandhedsværdien af relationen -1, og vi får

A=A-(-1)

eller simpelt hen

A=A+1

Hvis derimod A\$ ikke er "Z", bliver sandhedsværdien 0, og vi får

A=A-(0)

Pengene passer. Og dermed er de 23 linjer, der blev til 14, faktisk blevet til SYV! Nemlig

```
1 POKE650,128:POKE53280,1:53281,1:A=10:B=10:PRINT"h"
2 POKEA*40+B+983,160:POKEA*40+B+55255,C:GET
  A$:IFA$="F"THENGOSUB5
3 A=A+(A$="W")-(A$="Z"):B=B+(A$="A")-(A$="S"):A
  =A+(A=26)-(A=0):B=B+(B=41)-(B=0)
4 GOTO2
```

```

5 GETA$:IFA$=""THEN 5
6 IFASC(A$)< 49ORASC(A$)> 56THEN4
7 C=VAL(A$)-1:RETURN

```

Tryllekunsten består selvfølgelig i at blive af med IFerne ved at »indbygge« betingelsen i relationen eller udtrykket. Et udtryk af typen

$A = A - (A\$ = "Z")$

kaldes derfor også et BETINGET UDTRYK.

NB! Prøv

```

1 PRINT"̲":POKE650,128:POKE53280,0:POKE53281,0:
  TI$="000000":A=1000:B=2300
2 FORC=1984TO2023:POKEC,160:NEXT
3 PRINT"hf2TID"INT(TI/60):PRINT"FUEL"INT(A)"vm"
  :?"JET"INT(D):?"FART"INT(E)"vm"
4 PRINT"HØJDE"INT(B)"vm":IFB=0ANDE >=-1THE
  NPRINT"BLØD LANDING":POKE650,0:END
5 IFA <=0ORB=0ANDE < -1THENPRINT"NEDSTYR
  TNING":POKE650,0:END
6 GETA$:D=D+(D > 0)*.5-(A$ < > ""ANDD < 9):A=A
  -D/110:E=E+D/50-.05
7 POKE(23-INT((B-1)/100))*40+1020,32:B=B+E:IFB <
  0THENB=0
8 IFB > 2300THENB=2300
9 POKE(23-INT((B-1)/100))*40+1020,1:GOTO3

```

LEKTION 26

SKÆRMKODER

```
1 A=983:B=55255:C=2:D=2:E=24:F=38:POKE
  650,128:POKE53280,1:POKE53281,1:PRINT"h"
2 FORG=2TO24:FORH=2TO38STEP2:POKEA+G*40+
  H,46:POKEB+G*40+H,5:NEXT:NEXT
3 FORG=1TO39STEP2:FORH=1TO25:POKEA+G+H*
  40,160:POKEB+G+H*40,6:NEXT:IFG=1ORG=39
  TH5
4 FORI=1TO7:J=G+(INT(RND(1)*23)+2)*40:POKEA+J,
  46:POKEB+J,5:NEXT
5 NEXT:FORG=1TO25STEP24:FORH=1TO39:POKEA
  +G*40+H,160:POKEB+G*40+H,6:NEXT:NEXT
6 POKEA+C*40+D,32:GETA$
7 C=C+(A$="W"ANDPEEK(A+(C-1)*40+D) < 128)-
  (A$="Z"ANDPEEK(A+(C+1)*40+D) < 128)
8 D=D+(A$="A"ANDPEEK(A+C*40+D-1) < 128)-(A$
  ="S"ANDPEEK(A+C*40+D+1) < 128)
9 IFPEEK(A+C*40+D)=46THENK=K+10
10 POKEA+C*40+D,90:POKEB+C*40+D,2:POKEA+E
  *40+F,46:POKEB+E*40+F,2
11 E=E+((C < EORRND(1) < .5)ANDPEEK(A+(E-1)*40+
  F) < 128)
12 E=E-((C > EORRND(1) < .5)ANDPEEK(A+(E+1)*40+
  F) < 128)
13 F=F+((F > DORRND(1) < .5)ANDPEEK(A+E*40+F-1)
  < 128)
14 F=F-((F < DORRND(1) < .5)ANDPEEK(A+E*40+F+1)
  < 128):POKEA+E*40+F,88:POKEB+E*40+F,0
```

```
15 IFC=EANDD=FTHEN15
16 PRINT"hf7f9"K:GOTO6
```

A 7657
B 38377
C DIN LINJE
D DIN PLADS
E UHYRETS LINJE
F UHYRETS PLADS
G KONTROLVARIABEL
H KONTROLVARIABEL
I KONTROLVARIABEL
J KONTROLVARIABEL
K POINTS

1. Initialisation, hvid skærm, slet skærm.
2. Tegn og mal prikker.
3. Tegn og mal labyrint.
4. Tegn og mal passager.
5. Tegn og mal ydermur.
6. Slet dig, check for indtastning.
7. Du op eller ned?
8. Du til venstre eller højre?
9. Hvis prik spist, pointsforøgelse.
10. Tegn og mal dig, slet uhyre.
11. Uhyre op?
12. Uhyre ned?
13. Uhyre til venstre?
14. Uhyre til højre? Tegn og mal uhyre.
15. Fangede uhyret dig?
16. Skriv points

Vi har her næsten et »rigtigt« spil. Du kan styre din »brik« gennem labyrinten og spise de små grønne prikker, men ikke gå igennem mure. Hver spist prik giver ti points, og du

kan hele tiden se i øverste venstre hjørne, hvor mange du har fået. Samtidig skal du selvfølgelig undgå uhyret, der jager dig og samtidig efterlader et spor af røde prikker, der også giver points, hvis du spiser dem. I denne version giver de også ti points. Lav en version, hvor de giver tyve (peek prikkens farve). Eller lad små hjerter dukke op hist og her, som f.eks. giver 100 points. Eller lav endnu bedre versioner.

Lidt kedeligt er det nu immervæk. Du burde sige noget (altså din »brik«), mens du farer igennem labyrinten, og uhyret noget andet, mere faretruende. En prik skulle også give et lille gisp fra sig, når den mødte sit endeligt, et hjerte kunne give en fanfare, og dit uundgåelige endeligt en lille sørgelig melodistump. Så ville det straks begynde at ligne noget!

Faktisk skal vi senere beskæftige os med lyd. Før vi fanger an med noget helt nyt, er der dog som sædvanlig lige et par småting at samle op! Det bruger vi resten af kapitlet til!

Læg mærke til, hvordan vi fik labyrinten, nemlig med 160, altså $32+128$, altså $32 =$ mellemrum omvendt. Et blik på skærmmkodetabellen i appendikset bag i bogen vil vise dig, at den højeste skærmkode er 127, resten er »omvendte«.

Men sæt, vi nu vil bruge det andet karaktersæt, det med de små bogstaver? Prøv, mens du spiller labyrintspillet, at taste COMMODORE/SHIFT. Den røde rude skifter til et Z, »uhyret« til et X. Så nemt er det! Du kan naturligvis også POKE dig til skiftet. Så betyder

POKE53272,23 skift til karaktersæt 2

og

POKE53272,21 skift (tilbage) til karaktersæt 1

Du kan altså ikke have karakterer fra de to sæt på skærmen på én gang, ganske uanset, om du bruger tastaturet eller POKE.

Skriv

?PEEK(214)

Maskinen oplyser, på hvilken linje markøren befandt sig, da du »peekede«. Skriv nu

POKE214,100

Nu skal markøren være i linje 100, og det er selvfølgelig umuligt. Maskinen reagerer da også noget besynderligt. Prøv en LIST. Den er nervesammenbrudt nær.

Faktisk *kan* en datamat godt få nervesammenbrud. Rekreationsopholdet er dog kortvarigt og billigt. Sluk for maskinen og tænd igen, så er den i bogstavelig forstand så god som ny.

Symptomerne er lidt forskellige. Fælles for dem alle er det dog, at maskinen ikke længere reagerer fornuftigt på fornuftige ordrer, eventuelt slet ikke. Men nogen uhelbredelig sygdom er det altså ikke just.

NB! Prøv

```
1 REM HVIS DU UDSKIFTER 9 I 35 MED ET LA  
  VERE TAL, HAR DU EN BEDRE MODSTANDER  
2 POKE53280,0:POKE53281,1:FORA=1TO6:READA  
  $(A):NEXT  
3 A=100:B=100:C=1000:D=100:E=100:F=1000:G=9:  
  H=9:I=9:J=9  
4 K=1:L=L+1:PRINT"hf1f9"L"v. DAG":IFMTHENPRI  
  NT"f9FJENDENS SVAR:":PRINT"f9"A$(M)  
5 PRINT:PRINT"f7f9PATRIA"
```

```

6 PRINTA"FLY (PROD"G"v)":PRINTB"SKIBE (PROD
  "H"v)":PRINTC"SOLDATER"
7 PRINT:PRINT"f3f9HOSTILIA"
8 PRINTD"FLY (PROD"I"v)":PRINTE"SKIBE (PROD"
  J"v)":PRINTF"SOLDATER"
9 PRINT:PRINT"f51. BOMB LUFTHAVN":PRINT"2.
  BOMB HAVN":PRINT"3. SØSLAG"
10 PRINT"4. INVASION":PRINT"5. BOMB FLYFABR
  IK":PRINT"6. BOMB DOK"
11 PRINT"7. INGEN ORDRE":PRINT"f9ORDRE?"
12 GETA$:IFA$="7"THEN21
13 IFA$ < "1"ORA$ > "6"OR(A$ < "3"ORA$ > "4")AN
  DA=0ORA$="3"ANDB= 0ORA$="4"ANDC=0THE
  N12
14 ONVAL(A$)GOTO15,16,17,18,19,20
15 N=A:O=D:GOSUB35:A=N:D=O:ONKGOTO21,29
16 N=A:O=E:GOSUB35:A=N:E=O:GOTO21
17 N=B:O=E:GOSUB35:B=N:E=O:ONKGOTO21,29
18 N=C:O=F:GOSUB35:C=N:F=O:ONKGOTO21,29
19 N=A:O=I:GOSUB35:A=N:I=O:GOTO21
20 N=A:O=J:GOSUB35:A=N:J=O
21 P=0
22 P=P+1:IFP=25THEN31
23 M=INT(RND(1)*6)+1:IFM=1AND(A=0ORD=0)ORM
  =2AND(B=0ORD=0)ORM=3AND(B=0ORE=0)TH22
24 IFM=4AND(C=0ORF=0)ORM=5AND(D=0ORG=0)
  ORM=6AND(D=0ORH=0)THEN22
25 K=2:ONMGOTO15,26,17,18,27,28
26 N=B:O=D:GOSUB35:B=N:D=O:GOTO29
27 N=G:O=D:GOSUB35:G=N:D=O:GOTO29
28 N=H:O=D:GOSUB35:H=N:D=O
29 A=A+G:B=B+H:D=D+I:E=E+J:IFL < 25THENG=G
  -(G < 9):H=H-(H < 9):I=I-(I < 9):J=J-(J < 9)
30 GOTO4

```

```

31 G=A+B+C:H=D+E+F:PRINT"f3f9";:IFG=HTHENPR
    INT"FRED";
32 IFG > HTHENPRINT"PATRIA VANDT";
33 IFG < HTHENPRINT"HOSTILIA VANDT";
34 GOTO34
35 N=N-INT(RND(1)*0/9):IFN < 0THENN=0
36 O=O-INT(RND(1)*N/10):IFO < 0THENO=0
37 RETURN
38 DATALUFTHAVN BOMBET,HAVN BOMBET,SØS
    LAG,INVASION,FLYFABRIK BOMBET,DOK BOM
    BET

```

LEKTION 27

BIT-LOGIK

Skriv

? "2+2", 2+2, 2+2=4

Maskinen skriver 2+2, 4 og -1, nemlig *strengen* "2+2", *resultatet af udregningen* 2+2, og endelig *sandhedsværdien af relationen* 2+2=4. Tilsvarende giver

? 2+2=4 AND 2+2=5

0, eftersom AND kræver, at *begge* relationer skal være sande. Hvad med

? 2+2=4 OR 2+2=5

-1, fordi OR betyder, at det er nok, at *en* af relationerne (eventuelt begge) er sand. Endelig giver

? NOT 2+2=4

0, og

? NOT 2+2=5

-1

Vi kan udtrykke vores opdagelser i en SANDHEDSTABEL:

sand AND sand er SAND
sand AND falsk er falsk
falsk AND falsk er falsk

sand OR sand er sand
sand OR falsk er sand
falsk OR falsk er FALSK

NOT sand er falsk
NOT falsk er sand

Er der noget ved dette, der endnu ikke forekommer dig helt indlysende, så læs igen lektion 10 og lektion 25!

Prøv nu

?27AND50

Hvis formuleringen forekommer dig vanvittig, er resultatet det vel i endnu højere grad: 18! Hvad sytten har atten med syvogtyve og halvtreds at gøre? 27 fra 50 er ikke engang 18! Men prøv så at skrive de to tal op under hinanden:

27	0	0	0	1	1	0	1	1
50	0	0	1	1	0	0	1	0

18	0	0	0	1	0	0	1	0
----	---	---	---	---	---	---	---	---

Hvad betyder nu den sidste linje, der nærmest står som en slags facit under de to andre? Jo, det er sådan set, hvad det er. Maskinen har »ANDet« de to tal BIT FOR BIT på denne måde:

0 AND 0 (falsk AND falsk) er 0 (falsk)
0 AND 0 er 0
0 AND 1 (falsk AND sand) er 0

1 AND 1 (sand AND sand) er 1 (sand)

1 AND 0 er 0

0 AND 0 er 0

1 AND 1 er 1

1 AND 0 er 0

Det var da ikke så svært, vel? Næsten som i første klasse, når vi skulle regne $123+234$ ud! Lad os forsøge med OR:

?27OR50

59! Okay, vi siger

27 0 0 0 1 1 0 1 1

50 0 0 1 1 0 0 1 0

59 0 0 1 1 1 0 1 1

Ja, for

0 OR 0 (falsk OR falsk) er 0

0 OR 0 er 0

0 OR 1 (falsk OR sand) er 1

1 OR 1 (sand OR sand) er 1

1 OR 0 er 1

0 OR 0 er 0

1 OR 1 er 1

1 OR 0 er 1

Hvad med

?1000AND2000

1000 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0

2000 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0

960 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0

»Bevis« på samme måde

?1000OR2000

=2040! Hvad med

?100AND1000

100	0 0 0 0 0 0 0 0	0 1 1 0 0 1 0 0
1000	0 0 0 0 0 0 1 1	1 1 1 0 1 0 0 0
96	0 0 0 0 0 0 0 0	0 1 1 0 0 0 0 0

Hvad bliver

?100OR1000

Regn selv videre! Se det er BIT-LOGIK!

LEKTION 28

KARAKTERER

Skriv et A. Hvordan ved maskinen overhovedet, hvordan et A ser ud? Ja, det er en af de ting, fabrikken har fortalt den, som ligger i dens ROM.

Hvis du ser (meget) godt efter, vil du opdage, at et bogstav eller tegn på skærmen kan bestå af op til 8×8 små prikker, svarende til 8 bytes a 8 bits for en karakter. Det starter alt sammen i adresse 53248, således at f.eks. 53256-53263 er A. Lad os se, hvordan det går for sig.

Først er der dog et par ting, vi bør vide. Alt, hvad der kommer frem på skærmen, styres af den såkaldte Video Interface Chip eller »VIC-II«. Den finder vi i adresserne 53248-53294, og den må vi »koble ud« for at få fat i karaktergeneratoren. Nu kan maskinen selvfølgelig ikke længere reagere normalt på bla. tastatur, så det må vi også afbryde.

VIC-II bliver vi »af med« ved at poke i adresse 1, men da der er tale om en oplysning, der typisk kan klares af en enkelt bit (VIC-II slået til – VIC-II slået fra), står den alene i bit 2, idet vi traditionelt nummererer bits i en byte, så at f.eks. 137 bliver til

1	0	0	0	1	0	0	1
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Problemet bliver nu at ændre bit 2, uden at de oplysninger, som står i de andre bits, lider overlast. Det er nu, vi skal bruge vores bit-logik. Lad os sige, at der i adresse 1 står 55 eller

00110111

Vi skal altså have lavet 1-tallet (slået til) på plads 2 (tredjesidste) om til et 0 uden at pille ved de andre bits. Her kan vi bruge AND. Hvorfor? Jo, hvis vi ANDer 11111011, får vi

```
00110111
11111011
-----
00110011
```

idet kun to ettaller giver et ettal. Kan du se, at et hvilket som helst tal forbliver intakt på denne måde, bortset fra den enkelte ændrede bit, så vi uden betænkeligheder kan skrive

POKE1,PEEK(1)AND251

Vi skulle jo imidlertid også gerne kunne gå den anden vej, eftersom vi kun (meget) midlertidig kan undvære vores VIC-II. Nu bruger vi OR. Lad os altså sige, vi har

```
00110011
```

Vi ORer nu med 4 og får

```
00110011
00000100
-----
00110111
```

altså

POKE1,PEEK(1)OR4

Tastaturet afbryder vi i bit 0 af 56334, hvorfor de to nødvendige ordrer bliver

POKE56334,PEEK(56334)AND254

og (når vi skal »tænde« igen)

POKE56334,PEEK(56334)OR1

Prøv nu

```
1 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)
  AND251:FORA=1TO8:A(A)=PEEK(A+53255)
2 NEXT:POKE1,PEEK(1)OR4:POKE56334,PEEK(56
  334)OR1:FORA=1TO8:PRINTA(A):NEXT
```

Du får

24
60
102
126
102
102
102
0

I virkeligheden er disse tal jo oplagret i totalssystemet. Lad os lave dem om med programmet fra lektion 21,

```
1 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)
  AND251:FORA=1TO8:A(A)=PEEK(A+53255)
2 NEXT:POKE1,PEEK(1)OR4:POKE56334,PEEK
  (56334)OR1:FORA=1TO8:B=A(A)
3 FORC=7TO0STEP-1:IFB > =2pCTHENPRINT"1";:
  B=B-2pC:GOTO5
4 PRINT"0";
5 NEXT:PRINT:NEXT
```

hvilket giver

```
00011000
00111100
01100110
01111110
01100110
01100110
01100110
00000000
```

eller hvis vi udelader nullerne

```
11
1111
11 11
111111
11 11
11 11
11 11
```

Endnu tydeligere bliver det med

```
1 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)
  AND251:FORA=1TO8:A(A)=PEEK(A+53255)
2 NEXT:POKE1,PEEK(1)OR4:POKE56334,PEEK(563
  34)OR1:FORA=1TO8:B=A(A)
3 FORC=7TO0STEP-1:IFB > =2pCTHENPRINT
  "f9m";:B=B-2pC:GOTO5
4 PRINT"ø";
5 NEXT:PRINT:NEXT
```

Nu ville det jo være rart, hvis vi kunne skrive andet end A, og så må vi først være helt klare over, at karaktererne i karaktergeneratoren selvfølgelig er oplagret efter skærmkode.

Det hele starter med 53248, og vi kan få enhver karakter efter formelen

SKÆRMKODE*8+53248 OG 7 ADRESSER FREM

Da A er skærmkode 1, får vi følgende

1*8+53248 OG 7 ADRESSER FREM

eller 53256-53263. Vi kan nu udvide vores program til

```
1 INPUT"KODE";A:PRINT:POKE56334,PEEK(56334)
  AND254:POKE1,PEEK(1)AND251:FORB=1TO8
2 A(B)=PEEK(A*8+B+53247):NEXT:POKE1,PEEK(1)
  OR4:POKE56334,PEEK(56334)OR1
3 FORB=1TO8:C=A(B):FORD=7TO0STEP-1:IFC > =
  2pDTHENPRINT"f9m";:C=C-2pD:GOTO5
4 PRINT"ø";
5 NEXT:PRINT:NEXT:PRINT:RUN
```

NB! Prøv

```
1 POKE53280,1:POKE53281,1:A=1523:B=55795:C=
  160:PRINT"h"
2 D=INT(RND(1)*13):E=INT(RND(1)*13):F=
  INT(RND(1)*16):D=D*40:POKEA-D-E,C
3 POKEB-D-E,F:POKEA-D+E,C:POKEB-D+E,F:POK
  EA+D-E,C:POKEB+D-E,F:POKEA+D+E,C
4 POKEB+D+E,F:D=D/40:E=E*40:POKEA-D-E,C:
  POKEB-D-E,F:POKEA+D-E,C:POKEB+D-E,F
5 POKEA-D+E,C:POKEB-D+E,F:POKEA+D+E,C:
  POKEB+D+E,F:GOTO2
```

LEKTION 29

DEFINERBARE KARAKTERER

I lektion 28 havde vi et program, der gav os forstørrede udgaver af karaktererne, når vi indtastede skærmmkoden. Endnu bedre ville det selvfølgelig være, hvis vi kunne nøjes med at taste karakteren. Vi gør det på en måde, der samtidig introducerer tre systemvariabler (faktisk kender vi en af dem i forvejen), bl.a. fordi det demonstrerer, hvordan maskinen oplagrer sådanne, nemlig

209-210 BEGYNDELSEN PÅ MARKØRENS LINJE
211 MARKØRENS PLADS PÅ LINJEN

Hvis f.eks. markøren befinder sig på 10. linjes 10. plads, vil adresse 211 indeholde værdien 9. Maskinen regner med en linje 0 og en plads 0 som den første.

Hvad nu med 209-210? Jo, de skal fortælle maskinen, hvor den selv har oplagret plads 0 i den linje, hvor markøren befinder sig. Linje 10 (9 for maskinen) starter, som vi ved, i adresse $10 \cdot 40 + 1 + 983 = 1384$ (se lektion 23), og det er altså dette tal, de to systemvariabler skal opbevare.

To? Ja, for en enkelt kan jo kun oplagre tal op til 255. Vi vil nu finde, at værdierne i de to adresser er

209: 104

210: 5

Dette er for så vidt 256-talsystem og læses $104 + 256 \cdot 5 = 1384$. Generelt kan to nabo-bytes, der udtrykker ét tal, (som regel) læses

DEN FØRSTE BYTE + 256 * DEN ANDEN BYTE

Jeg siger »som regel«, fordi der kan være byttet om på de to. Formlen gælder dog universelt for systemvariabler, og det er trods alt der, vi får brug for den.

Det korte af det lange er, at vi nu altid kan finde markørens plads med

`PEEK(209)+PEEK(210)*256+PEEK(211)`

Vi starter nu med at tage den karakter ind, som maskinen skal forstørre, f.eks. med GETC\$. Derefter tilskrives vi markørens plads i skærmhukommelsen som værdi til variablen A. Endelig giver vi maskinen ordre til at skrive karakteren (PRINTC\$). Vi ved nu, at den vil blive skrevet der, hvor markøren stod (adresse A), og kan derfor med en ny peek finde frem til dens skærmkode.

Og så lidt pynt . . .

```

1 POKE53280,1:POKE53281,1:A$=f1f3f4f5f6f7f8c1c2c3
  c4c5c6c7c8":PRINT"h"
2 B$=MID$(A$,INT(RND(1)*15)+1,1):GETC$:IFC$=""
  THEN2
3 A=PEEK(209)+PEEK(210)*256+PEEK(211):PRINTC$:
  A=PEEK(A):PRINT"h"
4 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)
  AND251:FORB=1TO8
5 A(B)=PEEK(A*8+B+53247):NEXT:POKE1,PEEK(1)OR
  4:POKE56334,PEEK(56334)OR1
6. FORB=1TO8:C=A(B):FORD=7TO0STEP-1:IFC >=
  2pDTHENPRINTB$"f9m";:C=C-2pD:GOTO8
7 PRINT"ø";
8 NEXT:PRINT:NEXT:PRINT:GOTO2

```

Læg mærke til, at du godt kan indtaste flere karakterer ad

gangen, faktisk helt op til 10. Maskinen »gemmer« det, den får ind ved GET. Den kan endda finde på at skrive det ud, når programmet er kørt, så at du efter et spil kan ende med den lidt besynderlige aftakning WASZZZZ eller lignende. Det er en af grundene til, at vores spil gerne ender i en uendelig løkke . . .

Vi har nu fundet karaktergeneratoren. Den starter i adresse 53248, og eftersom hver karakter kræver otte adresser, og skærmkoder 128-255 er karakterer 0-127 omvendt (lektion 26), starter de omvendte karakterer i adresse $53248 + 128 * 8 = 54272$. Det andet karaktersæt får vi så fra adresse 55296, og dette samme i omvendt form fra 56320.

Nu ved vi, hvor vi kan finde karaktererne, men hvordan ved maskinen det? Prøv

?PEEK(53272)

Du får 21 eller 00010101. Det er bits 0-3 i denne byte, der fortæller maskinen, hvor den skal søge sine karakterer, i dette tilfælde (»normalt«) med startadresse i 53248. Tast nu COMMODORE/SHIFT og peek igen. Vi får 23 eller (0001) 0111. Det tal, maskinen læser, er nu ikke længere 5, men 7, og maskinen søger sine karakterer fra 55296. Resultatet er karaktersæt nummer to.

53272 findes faktisk i RAM, dvs. vi kan ændre dens værdi (lektion 21). Lad os prøve:

POKE53272,(PEEK(53272)AND240)+12

Alle karakterer på skærmen blev til »kinesisk«. Hvorfor nu det? Jo, for nu har vi fortalt maskinen, at den skal søge sine karakterer fra adresse 12288, og altså tolke, hvad der end står i disse adresser, som karakterer efter det system, vi lærte i forrige lektion. Resultatet bliver naturligvis pladder. Men sæt, vi nu i forvejen havde anbragt noget i adresserne,

der kunne tolkes som karakterer? Ja, så havde vi faktisk lavet vores egen karaktergenerator, vores egne karakterer.

Kan du se mulighederne? Vi kan få maskinen til at skrive arabisk! Eller vi kan lave figurer og billeder, der er detaljerede helt ned til den mindste prik, du nu kan se på skærmen. Vores spil vil forvandle sig fra at være en temmelig formålsløs flytten om med Aer og Oer til regulære små tegnefilm. Dette vil vi udforske grundigt i de følgende lektioner. Hvad står der nu i 12288—? Ja, adressen er en af de godt og vel 38K, som vi har til rådighed i vores programmer, variabler, osv. De går nemlig fra 2048 til 40959.

Problemet er bare, at vi hurtigt vil få slettet den nye karaktergenerator af de programmer og variabler, vi indtaster, når vi skal bruge den til noget. Hvordan undgår vi nu det?

Ja, hvordan undgår maskinen at skrive ind i området efter 40959? Jo, den læser hele tiden sin begrænsning i systemvariablerne 55 og 56. Bliv af med det kinesiske med RUN STOP/RESTORE. Tast nu

?PEEK(55)+PEEK(56)*256

Resultat: 40960: = HER BEGYNDER NOGET ANDET! Faktisk står det endnu et sted. Prøv

?PEEK(51)+PEEK(52)*256

Skriv endelig

?PEEK(56)

Vi har her fat i MSB, More Significant Byte, den mest betydningsfulde byte (55 er LSB, Less Significant Byte). 56 er mest betydningsfuld, fordi det tal, vi får ud af den (160), ikke betyder 160, men $160 * 256 = 40960$ eller 160 "256'ere". 55 indeholder kun enere, og i det foreliggende tilfælde indeholder den slet ikke noget. Siger vi nu

POKE52,48:POKE56,48

har vi ganske vist kun trukket 112 fra adressens værdi, men disse 112 betyder $112 * 256 = 28672$. Når maskinen herefter læser 52 og 56, vil den finde, at den har 28672 bytes mindre til sin rådighed. Nu falder 12288–40959 uden for den for programmer og variabler til rådighed stående hukommelse, og vi behøver ikke at frygte, at maskinen skal skrive over vores nyoprettede karaktersæt. POKE lidt frem og tilbage og tag en

?FRE(0)–(FRE(0)<0)*65538

for at kontrollere, at det passer. Vi har nu

1. Skaffet plads til vores karaktersæt.

Vi mangler at have

2. Oprettet vores nye karaktersæt

og

3. Fået maskinen til at tage sine karakterer i 12228– (med POKE53272,(PEEK(53272)AND240)+12).

Sådan er proceduren. Svært?

Jamen vi kan da starte med bare at »stjæle« maskinens eget karaktersæt og flytte det over til »os selv«. Prøv

- 1 POKE52,48:POKE56,48:POKE56334,PEEK(56334)
AND254:POKE1,PEEK(1)AND251
- 2 FORA=12288TO16383:POKEA,PEEK(A+40960):
NEXT
- 3 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
:POKE53272,(PEEK(53272)AND240)+12

LEKTION 30

NYT KARAKTERSÆT

I lektion 28 så vi, hvordan maskinen oplagrer sine karakterer, og i lektion 29, hvordan vi kan lave vores egen karakter-generator. Vi skulle nu have forudsætninger for at definere vore egne karakterer. Vi laver en lille frø:

Det bliver altså

00011000
00011000
01111110
11011011
00011000
00111100
00100100
01100110

eller



24
24
126
219
24
60
36
102

Vi vælger at undvære spar, hvilket betyder, at de otte værdier skal pokes ind i $12288 + 65 * 8 = 12808$ og 7 adresser frem eller 12808-12815.

Vi får nu »frø-programmet«

```
1 POKE52,48:POKE56,48:POKE56334,PEEK(56334)AND
  254:POKE1,PEEK(1)AND251
2 FORA=12288TO16383:POKEA,PEEK(A+40960):
  NEXT
3 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:
  POKE53272,(PEEK(53272)AND240)+12
4 FORA=12808TO12815:READB:POKEA,B:NEXT
5 DATA24,24,126,219,24,60,36,102
```

1. Diverse forberedelser.
2. Overførsel af karactersæt til RAM.
3. Tilbagevenden til normalt tilstand, oplysning om, hvor maskinen nu finder karactersættet.
4. Ny karakter.
5. Karakterens otte data.

Prøv at skrive ABC osv. Alt som sædvanlig. Tast nu spar!
Ikke sandt?

Når vi har »formlen«, er det i og for sig kun et spørgsmål om, hvilke værdier vi skal proppe i vores datalinjer. Det kan

selvfølgelig regnes ud med kvadreret papir og kugleramme, men hvorfor nu det, når vi har en datamat:

```
1 POKE53280,1:POKE53281,1:PRINT"h":FORA=1TO8
  :FORB=1TO8
2 POKEA*40+B+983,46:POKEA*40+B+55255,2:NEXT
  :NEXT:A=4:B=4:C=-1
3 POKEA*40+B+983,160:IFC=-1 THENPOKEA*40+B+
  983,46
4 POKEA*40+B+55255,0:GETA$:IFA$="F" THENC=-C
5 IFA$="L" THEN8
6 A=A+(A$="W")-(A$="Z"):B=B+(A$="A")-(A$="S")
  :A=A+(A=9)-(A=0):B=B+(B=9)-(B=0)
7 GOTO3
8 PRINT"ssssss"
9 FORA=1TO8:C=0:FORB=1TO8:C=C-(PEEK(A*40+B
  +983)=160)*2p(8-B):NEXT:PRINTC:NEXT
```

Hav dette program klart på bånd. Når du så skal bruge egne karakterer, loader du det og tegner din karakter. Du skifter fra tegn til slet og tilbage igen med F, og når du er færdig, bruger du L. De otte værdier vil så blive printet ud. Lad os se, hvordan det virker:

1. Hvid skærm, slet skærm.
2. Tegn og mal prikker, initialiser.
3. Tegn markør, hvis C=-1 slet markør.
4. Mal markør, check for indtastning; hvis F, skift fortegn på C.
5. Hvis L, gå til 8.
6. Markørbevægelse.
7. Om igen.
8. Syv linjer ned.
9. Udskriv værdier.

Prøv til sidst

```
1 POKE52,48:POKE56,48:POKE56334,PEEK(56334)
  AND254:POKE1,PEEK(1)AND251
2 FORA=12288TO16383:POKEA,PEEK(A+40960):
  NEXT
3 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:
  POKE53272,(PEEK(53272)AND240)+12
4 FORA=1TO11:READB:B=B*8+12288:FORC=BTOB
  +7:READD:POKEC,D:NEXT:NEXT
5 DATA7,126,64,64,64,64,64,64,,4,24,24,36,36,66,66,
  126,,21,24,36,66,90,66,36,24,
6 DATA12,24,24,36,36,66,66,66,,22,126,,,60,,,126,,18,
  126,36,36,36,36,36,36,
7 DATA19,126,64,32,16,32,64,126,,3,56,64,64,64,60,2,
  28,,6,8,28,42,42,42,28,8,
8 DATA23,42,42,42,28,8,8,8,,17,24,36,66,66,66,36,
  102,
```

Det kan ofte synes, som om maskinens mere avancerede funktioner kræver så enorme og indviklede programmer, at det næppe er umagen værd.

Men se nu på ovenstående: Formel, en indlæsningslinje og 4 datalinjer, og du har hele det græske alfabet til din rådighed!

TAST	GRÆSK BOGSTAV
A	ALPHA
B	BETA
G	GAMMA
D	DELTA
E	EPSILON
Z	ZETA
H	ETA
U	THETA

I	IOTA
K	KAPPA
L	LAMBDA
M	MU
N	NU
V	XI
O	OMICRON
R	PI
P	RHO
S	SIGMA
C	SIGMA (FINALT)
T	TAU
Y	UPSILON
F	PHI
X	CHI
W	PSI
Q	OMEGA

NB! Prøv

```

1 PRINT"h":POKE52,48:POKE56,48:POKE56334,
  PEEK(56334)AND254:POKE1,PEEK(1)AND251
2 FORA=12288TO12799:POKEA,PEEK(A+40960):
  NEXT
3 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:
  POKE53272,(PEEK(53272)AND240)+12
4 FORA=1TO5:READB:B=B*8+12288:FORC=BTOB+
  7:READD:POKEC,D:NEXT:NEXT
5 DATA27,255,129,165,129,165,129,165,129,28,36,
  126,255,36,36,255,,255
6 DATA29,240,40,20,251,20,40,240,,30,,,8,8,28,8,,,
  31,,64,42,21,85,161,165,129
7 POKE53280,1:POKE53281,1:FORA=2TO39:C=INT
  (RND(1)*14)+2:FORB=1TOINT(RND(1)*20)

```

```

8 POKE(25-B)*40+A+983,27:POKE(25-B)*40+A+
55255,C:NEXT:NEXT
9 FORA=1984TO2023:POKEA,28:POKEA+54272,5:
NEXT:A=1024
10 POKEA,32:A=A+1:POKEA,29:POKEA+54272,0:IF
PEEK(A+1)=27ORA=1903THEN20
11 GETA$:IFA$="m"ANDE=0THENB=A+40:E=1
12 IFE=1THENPOKEB,32:B=B+40:POKEB,30:POKEB+
54272,0:IFPEEK(B+40)=27THENGOSUB15
13 IFB > 1943THENPOKEB,31:POKEB+54272,
PEEK(B+54312):E=0
14 FORC=1TO50:NEXT:GOTO10
15 F=F+100:C=INT(RND(1)*15)+1
16 PRINT"h" MID$("f1f3f4f5f6f7f8c1c2c3c4c5c6c7c8",
C,1)F"m > mMETROPOLISm < "
17 POKEB,32:B=B+40:POKEB,30:POKEB+54272,2
18 IFRND(1) < .25ORB > 1903THENPOKEB,31:
POKEB+54272,PEEK(B+54312):E=0:RETURN
19 FORC=1TO50:NEXT:GOTO15
20 GOTO20

```


FJERDE DEL
Sprites, lyd m.m.

LEKTION 31

HØJOPLØSELIG GRAFIK

I begyndelsen af bogen lavede vi mange tabeller, lige fra 17-tabellen til rentesregning. En anden måde at fremstille den type materiale på er den grafiske fremstilling, kurven, som vi alle sammen kender. Skal vi lave noget lignende på COMMODORE 64, må vi have kontrol med den enkelte prik på skærmen. Det får vi ved at sætte den i »bit-mode«:

```
POKE53265,PEEK(53265)OR32
```

Tilbage igen kommer vi meget logisk med

```
POKE53265,PEEK(53265)AND223
```

Nu skal maskinen huske, hvilke prikker er tændt og slukket, og det må vi sætte noget hukommelse til side til:

```
POKE53272,PEEK(53272)OR8
```

Nu starter vores »bit-kort« i 8192. Skærmen består af 25 linjer à 40 karakterer à 8×8 prikker, hvor 8 prikker dækkes af en byte. Vores kort må altså gå frem til $8192 + 7999 = 16191$.

Hvad gør da vores normale skærmhukommelse 1024-2023. Jo, den kan få lov at bestemme farven. Lad os således sige, at der i en adresse i skærmhukommelsen står 137 eller

```
10001001
```

Ja så bliver prikken farve $1000=8=\text{orange}$ med farve $1001=9=\text{brun}$ baggrund. Vi kan nu begynde at opbygge et program:

```
1 PRINT"h":POKE53280,0:POKE53272,PEEK(53272)
  OR8:POKE53265,PEEK(53265)OR32
2 FORA=8192TO16191:POKEA,0:NEXT:FORA=1024
  TO2023:POKEA,1:NEXT
3 GOTO3
```

Kør dette, og en usynlig malerkost maler en fin hvid tavle til os. Nu skal vi bare finde ud af, hvordan vi tegner på den (med sort).

I virkeligheden er skærmen jo opdelt i linjer og pladser. Nu vil vi jo midlertid gerne »adressere« den enkelte bit nummereret 0–319X0–199. Hver prik angives så ved to tal (koordinater), en X-koordinat, der fortæller, hvor langt til højre vi finder prikken, og en Y-koordinat, der fortæller, hvor langt nede.

Hvis vi regner med en linje 0 og en plads 0, får vi, at

$$\text{LINJE}=\text{INT}(Y/8)$$

og

$$\text{PLADS}=\text{INT}(X/8)$$

Vi har nu karakteren, men vi skal også have den enkelte byte i karakteren, og den enkelte bit. BYTEN bliver nu

$$8192+\text{INT}(Y/8)*320+\text{INT}(X/8)*8+Y-\text{INT}(Y/8)*8$$

$Y-\text{INT}(Y/8)*8$ er i øvrigt det samme som $Y\text{AND}7$. Prøv med 137: $Y-\text{INT}(Y/8)*8$ bliver da 1, mens

```

10001001
00000111
-----
00000001

```

Det er jo kun resten efter otte-divisionen, der får lov til at blive stående, når der ANDes med 7 (111). Tilsvarende bliver BITen $7-(X \text{ AND } 7)$. Vi kan imidlertid ikke blot poke vores fundne byte 2pBIT, men må ORe for ikke at forstyrre andre eventuelt allerede tændte bits i byten. Vi kan nu prøve

```

1 INPUT X,Y:IF X<0 OR X>319 OR Y<0 OR Y>199 THEN
  EN1
2 PRINT"h":POKE53280,0:POKE53272,PEEK(53272)OR
  8:POKE53265,PEEK(53265)OR32
3 FOR A=8192 TO 16191:POKEA,0:NEXT:FOR A=1024
  TO 2023:POKEA,1:NEXT
4 A=8192+INT(Y/8)*320+INT(X/8)*8+(Y AND 7):
  POKEA,PEEK(A)OR2p(7-(X AND 7))
5 GOTO5

```

Traditionelt angiver i et koordinatsystem Y, hvor langt *oppe*, vi skal afsætte vores prik, så egentlig bør 4 hedde

```

4 A=8192+INT((199-Y)/8)*320+INT(X/8)*8+(199-
  Y AND 7):POKEA,PEEK(A)OR2p(7-(X AND 7))

```

PRINT"h" har naturligvis kun en æstetisk funktion.

Nu er der jo ikke meget sjov ved en prik. Kurverne og den egentlig *grafiske afbildning* tager vi fat på i lektion 32.

LEKTION 32

GRAFISK AFBILDNING

```
1 POKE53280,0:POKE53272,PEEK(53272)OR8:POKE
  53265,PEEK(53265)OR32
2 FORA=8192TO16191:POKEA,0:NEXT:FORA=1024
  TO2023:POKEA,1:NEXT:FORX=0TO 319
3 Y=X:Y=Y+(Y < 0)*Y+(Y > 199)*(Y-199)
4 A=8192+INT((199-Y)/8)*320+INT(X/8)*8+(199-
  YAND7):POKEA,PEEK(A)OR2p(7-(XAND7))
5 NEXT
6 GOTO6
```

Her har vi fået en kurve frem (egentlig en ret linje) ved hjælp af en FOR-NEXT-løkke. Ligningen $Y=X$ siger, at Y hele tiden skal være lige så stor som X , Y er en funktion af X . Andre funktioner kan afbildes på samme måde. Resten af linje 3 sikrer fladtoppedede kurver frem for kludder i adresserne. Lad os prøve

$$Y=\text{SQR}(X)$$

som vi altså blot propper ind i linje 3, hvor der nu står $Y=X$. Vi får nærmest en flad trappe. En graf, der ligner en kurve mere og desuden fylder skærmen bedre ud, får vi med

$$Y=\text{SQR}(X)*10$$

og STEP5 på FOR-NEXT-løkken. Personlig foretrækker jeg dog en sammenhængende graf, hvorfor det sidste nummer

ikke vil blive brugt i senere eksempler. Lad os altså droppe STEP og forsøge os med

$$Y=Xp2$$

Her vil

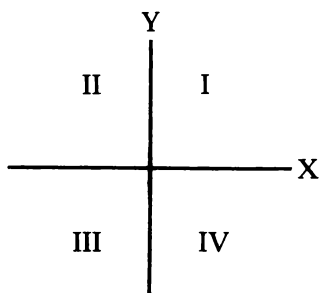
$$Y=INT(Xp2/500)$$

formodentlig være at foretrække.

Men hvad med negative værdier af X? Og af Y? Nåja, af Y vil der ikke være nogen, for minus gange minus giver plus. Herimod giver

$$Y=Xp3$$

sådanne værdier. Dem kan vores koordinatsystem ikke klare. Et »rigtigt« koordinatsystem ser da også omtrent sådan ud



De fire områder betegnet med romertal er de fire »kvaranter«. I er det, vi havde i forvejen. II medtager også negative værdier af X, IV klarer de negative af Y, og i III er både X og Y negative. Sådan et komplet med »akser« kan vi også lave:

```

1 POKE53280,0:POKE53272,PEEK(53272)OR8:POKE
  53265,PEEK(53265)OR32
2 FORA=8191TO16191:POKEA,0:NEXT:FORA=1024
  TO2023:POKEA,1:NEXT:FORA=0TO39
3 POKEA*8+12035,255:NEXT:FORA=0TO24:FORB=
  0TO7:C=A*320+B+8352:POKEC,PEEK(C)OR64
4 NEXT:NEXT:FORX=-160TO159:Y=Y+(Y<-100)*
  (Y+100)+(Y>99)*(Y-99):P=X+160
5 Q=Y+100:Y=X:A=8192+INT((199-Q)/8)*320+INT
  (P/8)*8+(199-QAND7)
6 POKEA,PEEK(A)OR2p(7-(PAND7)):NEXT
7 GOTO7

```

De to »variable« i dette standardprogram er

FORX=-160TO159

i linje 4, der kan tilføjes forskellige STEP (grafene kan nemlig også være for usammenhængende, så den skal »tætnes« med STEP mindre end 1), og

$Y=X$

i linje 5, der er funktionen, der skal afbildes. Prøv nu

$Y=Xp^2$

eller bedre

$Y=Xp^2/100$

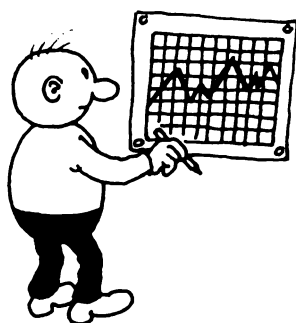
og

$Y=Xp^3/10000$

og

$$Y=\text{SIN}(X/50)*50$$

Du skulle nu være rustet til at gå på opdagelse i grafisk af-
bildning . . .!



LEKTION 33

SPRITES

En SPRITE kan nærmest beskrives som en definerbar karakter i overstørrelse. Mens en karakter kun kan være 8×8 prikker, er SPRITEn 24 prikker bred og 21 prikker høj. Som du måske kan regne ud, skal vi bruge $3 \times 21 = 63$ bytes til at definere sådan en, nemlig

0	1	2
3	4	5
6	7	8
.....		
57	58	59
60	61	62

Vi kan nu gå i gang med at tegne vores SPRITE på kvadreret papir, dele den op i 63 dele, lave prikkerne om til 1-taller, lave 1-tallerne om til titalssystemet og hæve vores folkepension. Dem, der finder det uhygiejnisk, at skægget vokser fast i datamaten, kan naturligvis lade et program tage sig af det grove arbejde, f.eks.

```
1 DIMA(62):POKE53280,1:POKE53281,1:PRINT"h"
  :FORA=1TO21:FORB=1TO24
2 POKEA*40+B+983,46:POKEA*40+B+55255,2:NEXT
  T:NEXT:A=11:B=12:C=-1
3 POKEA*40+B+983,160:IFC=-1THENPOKEA*40+B+
  983,46
4 POKEA*40+B+55255,0:GETA$:IFA$="F"THENC=-C
```

```

5 IFA$="L"THEN8
6 A=A+(A$="W")-(A$="Z"):B=B+(A$="A")-(A$="S")
  :A=A+(A=22)-(A=0):B=B+(B=25)-(B=0)
7 GOTO3
8 FORA=0TO20:FORB=0TO2:C=0:FORD=1TO8:C=C
  -(PEEK(A*40+B*8+D+1023)=160)*2p(8-D)
9 NEXT:A(A*3+B)=C:NEXT:NEXT:PRINT"hf3":FORA
  =0TO62:PRINTA(A)"v,";:NEXT:PRINT"vm"

```

Du kan have programmet på bånd og tage det ind, når du skal lave SPRITES. Når du kører det, vil du se 24X21 røde prikker og en blinkende sort markør, som du flytter med WASZ. Når du vil have farve på, taster du F, og et F til gør blyanten til viskelæder. Når du er færdig, taster du L. Du vil nu få de 63 værdier printet ud med komma imellem, så de er lige til at proppe ind i et par DATA-linjer. Lad os prøve at lave en SPRITE sammen, en simpel en, vi kan blive enige om.

Vi kan f.eks. lave en spiral ved at slå farven til, gå 1 op, 2 til højre, 3 ned, 4 til venstre, 5 op, etc., til spiralen når kant (vi kommer til at ende i øverste række, plads nummer 2 fra venstre). Tast nu L, og vi får værdierne

```

64,0,0,95,255,252,80,0,4,87,255,244,84,0,20,85,255,
212,85,0,84,85,127,84,85,65,84,85,93,84,85,85,84,85,69,
84,85,125,84,85,1,84,85,255,84,84,0,84,87,255,212,80,0,
20,95,255,244,64,0,4,127,255,252

```

Men hvor skal vi gøre af dem? Vi skal bruge 63 bytes, men 64 er selvfølgelig lettere for maskinen at regne med. I alle tilfælde skal maskinen selv vide, hvor den finder sine SPRITES. Du kan arbejde med 8 sprites på en gang, nummereret 0-7, men du må fortælle maskinen i adresse SPRITE-NUMMER+2040, hvor den skal finde dem. Prøv

```

1 POKE53280,1:POKE53281,1:PRINT"h":POKE2040,
  192
2 FORA=12288TO12350:READB:POKEA,B:NEXT
3 DATA64,,,95,255,252,80,,4,87,255,244,84,,20,85,255
  ,212,85,,84,85,127,84,85,65
4 DATA84,85,93,84,85,85,84,85,69,84,85,125,84,85,1,84
  ,85,255,84,84,,84,87,255
5 DATA212,80,,20,95,255,244,64,,4,127,255,252

```

Som du ser, ganger maskinen tallet i 2040 med 64. Kør programmet. Der sker ikke noget. Vi er nemlig ikke helt færdige endnu.

For det første skal vi have »tændt« vores sprite. »Kontakten« sidder i 53269, eller rettere de 8 kontakter. Skal du f.eks. tænde sprite 3, ja så slår du bit 3 til: POKE53269,8. Skal du have tændt 4 og 6, får vi selvfølgelig POKE53269,80. Det er jo

01010000

Her skal vi selvfølgelig blot have

POKE53269,1

Vi skal også have farve på spriten. Vi tager sort og siger

POKE53287,0

Som du vil forstå, styres spriternes farver af

SPRITE-NUMMER+53287

Endelig mangler vi at placere vores sprite på skærmen. Den kan selvfølgelig have $25 \times 8 = 200$ positioner i lodret retning, men da byten kan opbevare indtil 255, »starter« spriten fak-

tisk oven over skærmen med værdien 0 og kommer først til syne ved 30. Vi kan sige

POKE53249,100

Vi får denne Y-position med

SPRITE-NUMMER*2+53249

Med X-positionen er situationen modsat, 255 er simpelt hen ikke nok, vi skal i al fald bruge $40 \cdot 8 = 320$ af dem eller simpelt hen 9 bits. Den manglende finde vi i MOST SIGNIFICANT BIT X POSITION REGISTER (XMSB), og der er selvfølgelig plads til den manglende bit for alle 8 sprites. XMSB finder vi i 53264, mens resten af X-positionen ligger i

SPRITE-NUMMER*2+53248

Lad os også sætte den til 100 og få

POKE53248,100

Vi har nu programmet

```
1 POKE53280,1:POKE53281,1:PRINT"h":POKE2040,
  192:FORA=12288TO12350:READB:POKEA,B
2 NEXT:A=53248:POKEA+21,1:POKEA+39,0:POKEA
  ,100:POKEA+1,100
```

+ datalinjerne. Eller skift datalinjerne ud med

```
3 DATA26,126,88,26,90,88,30,255,120,30,231,120,6,126
  ,96,6,66,96,6,126,96,6,24
4 DATA96,7,255,224,7,255,224,,66,,,90,,,94,,,90,,,66,,15
  ,255,240,15,255,240,15
5 DATA255,240,12,,48,60,,60,60,,60
```

LEKTION 34

SPRITES I BEVÆGELSE

```
1 POKE53280,1:POKE53281,1:PRINT"h":POKE2040,
  192
2 FORA=12288TO12350:READB:POKEA,B:NEXT:
  A=53248:POKEA+21,1:POKEA+39,2
3 DATA26,126,88,26,90,88,30,255,120,30,231,120,6,126
  ,96,6,66,96,6,126,96,6,24
4 DATA96,7,255,224,7,255,224,,66,,,90,,,94,,,90,,,66,,15
  ,255,240,15,255,240,15
5 DATA255,240,12,,48,60,,60,60,,60
6 FORB=0TO255:POKEA,B:POKEA+1,B:NEXT:GO
  TO6
```

Programmets første fem linjer er standard definition af en sprite. Alle ordrene er gennemgået i detalje i forrige lektion. I denne skal vi se lidt på, hvad vi kan lave med den færdige sprite, og 1-5 er underforstået i de følgende eksempler.

Den nuværende 6 er hverken særlig vanskelig eller særlig spændende. Vi kunne endda have nøjedes med

```
6 POKEA,160:FORB=0TO255:POKEA+1,B:NEXT:GO
  TO6
```

Nu ramler vores Commodore-trold ned fra himlen og forsvinder i jorden. Lidt sværere bliver det at få ham fra venstre til højre. Prøv

```
6 POKEA+1,100:FORB=0TO255:POKEA,B:NEXT
```

Trolden forsvinder ikke til højre. Vi mangler XMSB og må have

```
6 POKEA+1,100:FORB=0TO400:C=INT(B/256):D=B  
-C*256:POKEA,D:POKEA+16,C:NEXT
```

Havde der været tale om sprite 5, skulle vi selvfølgelig have haft

```
POKEA+16,32
```

eller 2p5

De mange udregninger har selvfølgelig sinket trolden en del. Man må finde det smarteste i hvert enkelt tilfælde (i stedet for den ovenstående langsomme universalløsning), i dette tilfælde måske simpelt hen

```
6 POKEA+1,100:FORB=0TO255:POKEA,B:NEXT:POK  
EA+16,1:FORB=0TO100:POKEA,B:NEXT
```

Vi skal ikke gå i detaljer med hensyn til styringen af spriten ved hjælp af RND(1), tastatur osv., da de samme regler gælder som for flytning af almindelige karakterer.

Derimod er der ting, vi kan gøre med vores sprite, som vi ikke kan gøre med karakterer. Du har formodentlig allerede bemærket spritens fuldstændig glidende bevægelse. Prøv nu

```
6 POKEA,100:B=A+29:C=50:D=1  
7 C=C+D:IFC=50THEND=-D:POKEB,PEEK(B)OR1  
8 IFC=200THEND=-D:POKEB,PEEK(B)AND254  
9 POKEA+1,C:GOTO7
```

Som det vil fremgå, betyder den relevante bit tændt i 53277, at spriten skal udvides til det dobbelt i bredden. Prøv at sætte B til A+23 i stedet!

Lad os prøve engang at lege med to sprites på en gang. For enkeltheds skyld bruger vi de samme data:

```
1 POKE53280,1:POKE53281,1:PRINT"h":POKE2040,
  192:POKE2041,192
2 FORA=12288TO12350:READB:POKEA,B:NEXT:A=
  53248:POKEA+21,3:POKEA+39,2:POKEA+40,6
6 FORB=0TO3:POKEA+B,(B+1)*50:NEXT
```

Vi har nu en rød trold og en blå trold. Vi bevarer igen 1-5, men udskifter 6 med

```
6 DEFFNA(X)=INT(RND(1)*256):B=FNA(X):C=FNA
  (X):D=FNA(X):E=FNA(X):F=1:G=1
7 B=B+F:C=C+G:IFB=0ORB=255ORRND(1)<.01
  THENF=-F
8 IFC=0ORC=255ORRND(1)<.01THENG=-G
9 GETA$:IFA$<>" "THEND=D+(D>B)-(D<B):E
  =E+(E>C)-(E<C):H=H+1
10 POKEA,B:POKEA+1,C:POKEA+2,D:POKEA+3,E:
  PRINT"hf6"H:H=H+1:GOTO7
```

Kør programmet, og den røde og blå trold kommer til syne. Den røde vandrer lidt rundt, mens den blå står bomstille, indtil du berører mellemrumstasten. Så længe du holder den nede, nærmer den blå trold sig den røde trold, men samtidig tæller den blå trolds grønne kalorieforbrug dobbelt så hurtigt. Hvornår og hvor meget skal du taste?

Selv om dette spil næppe kommer til at udkonkurrere PACKMAN, ville der dog være en vis ræson i det, hvis spil og tæller standsede, når den blå trold fangede den røde. Hvordan konstaterer maskinen dette? Vi kunne have


```

10 POKEA,B:POKEA+1,C:POKEA+2,D:POKEA+3,E
   :IFB=DANDC=ETHEN10
11 PRINT"hf6"H:H=H+1:GOTO7

```

men det kan også gøres på en anden måde. Skriv

```

10 POKEA,B:POKEA+1,C:POKEA+2,D:POKEA+3,E:
   IFPEEK(A+30)=3THEN10

```

Læg mærke til, at figurerne nu blot behøver at berøre hinanden. En hvilken som helst overlapning mellem en sprite og en anden sprite »tænder« den pågældende bit i 53278. Tallet 3 angiver følgelig, at 0 og 1 er stødt sammen. Skift tilbage til den version, hvor figurerne dækker hinanden. Faktisk dækker den røde den blå. En sprite med mindre »nummer« dækker altid en med større. Prøv

```

11 PRINT"f6"H,PEEK(A+30),PEEK(A+31):H=H+1:GOTO7

```

Hvis du lader figurerne vandre lidt rundt, vil du finde, at overlapning mellem en sprite og baggrunden, i dette tilfælde de tre tal, manifesteres i 53279. PEEK(53279)=27=00011011 betyder altså, at sprites 0, 1, 3 og 4 er gerådet i karabolage med baggrunden.

Bemærk også, at begge figurer passerer over baggrunden. Denne prioritering kan ændres i 53275, idet en tændt bit betyder, at den tilhørende sprite passerer bag baggrunden. Tilføj i linje 1

```
POKE53275,2
```

og den røde figur vil stadig dække baggrunden, mens den blå vil blive dækket af den (tallene).

NB! Prøv

```
1 PRINT"h":POKE53269,255:POKE53280,11:POKE
  53281,11:FORA=0TO7:POKEA+2040,192
2 X(A)=127:Y(A)=127:NEXT:FORA=12288TO12796:
  POKEA,255:NEXT
3 FORA=0TO7:X(A)=X(A)+(RND(1)<.5)-(RND(1)<.5)
  :Y(A)=Y(A)+(RND(1)<.5)-(RND(1)<.5)
4 X(A)=X(A)+(X(A)=255)-(X(A)=0):Y(A)=Y(A)+(Y(A)=25
  5)-(Y(A)=0)
5 POKE53248+A*2,X(A):POKE53249+A*2,Y(A):
  POKEA+53287,INT(RND(1)*16):NEXT
6 POKE53271,INT(RND(1)*256):POKE53277,INT(RND
  (1)*256):GOTO3
```

LEKTION 35

SPRITES I SPIL

```
1 PRINT"h":FORA=1TO25:POKEA*40+1015,66
:POKEA*40+55287,2:NEXT
2 FORA=7TO0STEP-1:FORB=1TO30:POKEA*120+
  B+1144,45:POKEA*120+B+55416,11:NEXT
3 POKEA*120+1144,(8-A)+176:POKEA*120+
  55416,14:NEXT
4 A=53248:POKEA+21,255:POKEA+23,255:POKEA
  +29,255:POKEA+32,13:POKEA+33,1
5 FORB=0TO7:POKE2040+B,192:POKEA+B+39,B+
  2:POKEA+B*2+1,209-B*22:NEXT
6 FORB=12288TO12350:READC:POKEB,C:NEXT:
  TI$="000000"
7 FORB=0TO7:POKEA+B*2,A(B):A(B)=A(B)+INT
  (RND(1)*3):IFA(B)>226ANDD=0THENGOSUB11
8 IFA(B)>252THEN8
9 NEXT:IFD=0THENPRINT"hf9fl"TI$
10 GOTO7
11 PRINT"hf9fløøøøøøøøøø"B+1"VANDT":D=1:RETURN
12 DATA,16,,1,240,,,112,,1,240,,,16,,,16,,,16,,,16,,,
  16,,,16,,,16,,,254,,1
13 DATA255,,3,255,128,7,255,206,15,255,234,31,255,
  255,63,255,252,255,255,255,5
14 DATA1,64
```

- 1 Slet skærm, tegn og mal målstreg.
- 2 Tegn og mal baner.
- 3 Tegn og mal banenumre.

- 4 Tænd alle sprites, forstør alle sprites i begge retninger, lysegrøn kant, hvid skærm.
- 5 Læs alle sprites i 12288, spildpaddernes farver og linjer.
- 6 Læs sprite-data, lav sprites, nulstil ur.
- 7 Skildpaddernes pladser, flytning, målstreg nået?
- 8 Højrekant nået?
- 9 Tid.
- 10 Om igen.
- 11 Resultat.
- 12-14 Sprite-data.

Her er vores viden om sprites anvendt i et simpelt program. Ved at gennemgå det, skulle du få en fornemmelse for, hvordan sprites virker, så du selv kan bruge dem i (langt bedre) programmer.

```

1 POKE650,128:PRINT"h":FORA=1824TO2023
  :POKEA,160:POKEA+54272,5:NEXT
2 FORA=1TOINT(RND(1)*100)+100:B=INT(RND(1)
  *800)+1024:POKEB,81:POKEB+54272,1
3 NEXT:FORA=1TOINT(RND(1)*20)+20:B=INT(RND
  (1)*200)+1824:POKEB,160
4 POKEB+54272,15:NEXT:POKE2040,192:POKE
  2041,193:A=53248:POKEA+21,3:POKEA+23,3
5 POKEA+27,1:POKEA+29,3:POKEA+32,6:POKEA
  +33,14:POKEA+39,2:POKEA+40,0
6 FORB=12288TO12350:POKEB,0:NEXT:FORB=
  12309TO12328:READC:POKEB,C:NEXT
7 FORB=12352TO12414:READC:POKEB,C:NEXT:TI$
  ="000000"
8 B=150:C=50:D=150:E=125:F=2:G=2:H
  =1
9 POKEA,B:POKEA+1,C:POKEA+2,D:
  POKEA+3,E

```

```

10 IFABS(B-D+1) < 2ANDABS(C-E-3) < 2ANDH
   =1ANDC < 190THEN20
11 IFABS(B-D+1) < 2ANDABS(C-E+1) < 2ANDH
   =-1ANDC < 190THEN21
12 B=B+F:C=C+G:IFB=0ORB=254ORRND(1) < .
   05ANDH=1THENF=-F
13 IFC=0ORC=254ORRND(1) < .05ANDH=1THENG=
   -G
14 GETA$:D=D+((A$="A")-(A$="S"))*3:E=E+
   ((A$="W")-(A$="Z"))*3
15 D=D+((D=255)-(D=0))*3:E=E+((E=254)-(E=2))
   *3:PRINT"hf9f7"TI$mNEDSKUdT"J
16 IFRND(1) < .05THENH=-H:POKEA+39,4-H*2
17 IFH=-1THENF=((B > D-1)-(B < D-1))*2:G=((C > E-
   1)-(C < E-1))*2
18 IFTI$ > "000500"THEN18
19 GOTO9
20 POKEA+33,6:FORI=1TO1000:NEXT:POKEA+
   33,14:J=J+1:GOTO8
21 POKEA+33,2:FORI=1TO1000:NEXT:POKEA+
   33,14:PRINT"f9f7DU BLEV SKUdT NED"
22 GOTO22
23 DATA255,255,255,17,36,136,17,36,136,17,60,136,
   255,255,255,,36,,,36
24 DATA127,255,252,64,16,4,64,16,4,64,16,4,64,16,4,
   67,255,132,66,16,132,66,16
25 DATA132,66,16,132,66,16,132,127,255,252,66,16,
   132,66,16,132,66,16,132,66,16
26 DATA132,67,255,132,64,16,4,64,16,4,64,16,4,127,
   255,252,,,,

```

- 1 Repeter på alle taster, slet skærm, tegn og mal jord.
- 2 Tegn og mal skyer.
- 3 Tegn huse.

- 4 Mal huse, læs sprite 0 i 12288, sprite 1 i 12352, tænd sprites 0-1, 0-1 dobbelt bredde.
- 5 Sprite 0 (fly) passerer bag baggrunden (skyer og horisont), 0-1 dobbelt højde, mørkeblå kant, lyseblå skærm, sprite 0 rød, sprite 1 sort (sigtekorn).
- 6 Ryd sprite 0, læs og lav sprite 0.
- 7 Læs og lav sprite 1, nulstil ur.
- 8 Udgangspositioner for fly og sigtekorn, flyets hastighed, fly i defensiv.
- 9 Anbring fly og sigtekorn.
- 10 Hvis fjendtligt flys cockpit ramt, mens det er i defensiven og over horisonten, fly nedskudt.
- 11 Hvis du kommer i fjendtligt flys kanonsigte, mens det skyder og er over horisonten, er du skudt ned.
- 12 Flytning af fly, fjendtligt flys undvigelsesmanøvre.
- 13 Undvigelsesmanøvre II.
- 14 Sigtekorns flytning.
- 15 Tid, angivelse af nedskudte fly.
- 16 Ændring af fjendtligt flys taktik og farve, rød i undvigelsesmanøvre, blå i angreb.
- 17 Angrebstaktik.
- 18 Er tid udløbet?
- 19 Om igen.
- 20 Visuel effekt, pointstælling, ny omgang.
- 21 Visuel effekt, nedskydning.
- 22 Slut.
- 23 Fly-sprite-data.
- 24-26 Sigtekorn-sprite-data.

- A 53248.
- B Flys X-koordinat.
- C Fly Y.
- D Sigtekorn X.
- E Sigtekorn Y.
- F Flys X-flag.

- G Flys Y-flag.
- H Flys taktik-flag.
- I Kontrolvariabel for visuelle effekter.
- J Antal nedskudte fly.

Når du har forstået skildpadderne, kan du tage hul på ovenstående, der er et »rigtigt« spil. Når du også forstår det i en sådan grad, at du kan lave om på det (udbygget landskab, flere fly, ændret taktik, flere effekter, etc.), skulle du vide nok om sprites til at lave dine egne spil og programmer. God fornøjelse!

LEKTION 36

TONEN



- 1 A=54272
- 2 POKEA+24,15
- 3 POKEA+5,68
- 4 POKEA+6,68
- 5 POKEA,24:POKEA+1,14
- 6 POKEA+4,33
- 7 FORB=1TO1000:NEXT
- 8 POKEA+4,32

Som du vil forstå, når du kører det, har dette komplicerede program kun til formål at producere en enkelt tone (et A). Jeg behøver imidlertid formodentlig heller ikke fortælle dig, at dette er grund- og engangsindstillinger, hvad enten du spiller 2 eller 20 toner. Det er faktisk kun linje 5, der angiver selve tonen. Lad os prøve at se på det, linje for linje.

1. Dette pladsbesparende trick kender vi fra før.
2. Her bestemmes lydstyrken (0-15).
- 3.-4. En tone har imidlertid et *forløb*. Dette bestemmes i 3 og 4.

For at forstå dette, kan vi inddele forløbet i 4 faser:

- A Attack: Tonen stiger i lydstyrke fra 0 til sit højeste niveau. Ved at påvirke attack ændrer vi den hastighed, hvormed dette sker.
- B Decay: Tonen falder i lydstyrke fra sit højeste niveau til et midter-niveau. Ved at påvirke decay ændrer vi den hastighed, hvormed dette sker.

C Sustain: Dette er midter-niveauet. Ved at påvirke sustain ændrer vi middellydstyrken.

D Release: Tonen falder i lydstyrke fra sit midter-niveau til 0. Ved at påvirke release ændrer vi den hastighed, hvormed dette sker.

De fire faser kontrolleres af linje 3 og 4, nemlig således:

	ATTACK	DECAY	
54277	0100	0100	= 01000100 = 64
	SUSTAIN	RELEASE	
54278	0100	0100	= 01000100 = 64

Alle fire værdier er middelværdier. Højeste værdi er 1000, en mindre 0010, og mindste 0001. Allerhøjest ville selvfølgelig 1111 være. Eksempelvis ville højeste ATTACK og laveste DECAY altså være

POKE54277,241 (=11110001)

Prøv

4 POKEA+6,79

svarende til

0100 1111

altså samme sustain, men en meget lang release. Tonen er meget længe om at »klinge ud«. Skift så til

4 POKEA+6,65

eller ligefrem

4 POKEA+6,64

(ingen release). Tonen standser brat (efter sustain).

5. Selve tonen. Den værdi, der pokes ind, er i virkeligheden $24+14 \cdot 256$ (se lektion 29!). Tallet er altså 3608.
6. Her bestemmes selve lydbølgens form. Vi har 4 muligheder:

A TREKANT (POKEA+4,17)

B SAVTAK (POKEA+4,33)

C FIRKANT (POKEA+4,65)

D STØJ (POKEA+4,129) – til lydeffekter

For at få firkanten, må vi imidlertid også opgive pulsbredde. Prøv

3 POKEA+3,1:POKEA+5,68

6 POKEA+4,65

7 FORB=1TO1000:NEXT

8 POKEA+4,64

7. Så længe spiller tonen.
8. A+4-værdien en tak ned, og tonen slukker.

Prøv at eksperimentere videre med disse begreber, før du går videre.

LEKTION 37

MUSIK

```
1 A=54272:POKEA+24,15:POKEA+5,15:POKEA+
  6,240
2 FORB=1TO74:READC,D:E=INT(C/256):F=C-E*
  256:POKEA,F:POKEA+1,E
3 POKEA+4,33:FORC=1TOD*200:NEXT:POKEA+
  4,32:FORC=1TO10:NEXT:NEXT
4 DATA4817,2,6430,2,3215,2,4050,2,4817,2,6430,3,72
  17,1,8101,2,8583,2,8101,4
5 DATA7217,4,6430,4,,1,8101,2,8101,1,7217,1,6430,1,
  6069,1,7217,1,6430,1,6069,1
6 DATA5407,1,6069,3,6430,1,7217,2,7217,2,9634,2,72
  17,2,6069,2,4817,2,7217,3
7 DATA3608,1,3608,2,9634,2,9634,2,9094,1,8101,1,
  7217,2,8101,2,8101,2,7217,1
8 DATA6430,1,6069,2,6430,2,6069,4,5407,4,4817,4,,
  2,4817,2,6430,2,4817,2,5407,2
9 DATA5728,2,5407,3,5728,1,5407,2,5407,2,8583,3,81
  01,1,7217,3,6430,1,6069,2
10 DATA5407,2,4817,2,4817,2,6430,4,7217,4,8101,3,8
  583,1,9634,2,8583,2,8101,4
11 DATA7217,4,6430,4
```

- 1 A=54372, lydstyrke, attack-decay, sustain-release.
- 2 Læs tone og varighed, fordel toneværdi på 54272 og 54272.
- 3 Vælg bølgeform, hold tone, sluk tone, pause, om igen.
- 4-11 Data.

Som det vil fremgå, angives en pause i melodi med et 0 (underforstået i DATA) fulgt af tallet for pausens varighed (1/8-node=1). Tempo afgøres af tallet, vi ganger D med i linje 3. Prøv at skifte 200 ud med 100! Med 50!

Nu, hvor du har en rigtig melodi at eksperimentere med, kan du selvfølgelig også langt bedre danne dig et indtryk af virkningen af forandring af de forskellige værdier, attack, decay, etc. Prøv

```
3 POKEA+4,17:FORC=1TOD*300:NEXT:POKEA+
4,16:FORC=1TO10:NEXT:NEXT
```

eller

```
3 POKEA+4,65:FORC=1TOD*75:NEXT:POKEA+4,64
:FORC=1TO10:NEXT:NEXT
```

I det sidste tilfælde skal du selvfølgelig huske at angive puls-bredde. Du kan til 1 tilføje

```
POKEA+3,1
```

Eller prøv

```
1 A=54272:POKEA+24,15:POKEA+5,255:POKEA+6,
255:POKEA+12,15:POKEA+13,240
2 FORB=1TO74:READC,D:E=C*2:F=INT(C/256):G=C
-F*256:H=INT(E/256):I=E-H*256
3 POKEA,G:POKEA+1,F:POKEA+7,I:POKEA+8,H:
POKEA+4,17:POKEA+11,33:FORC=1TOD*200
4 NEXT:POKEA+4,16:POKEA+11,32:FORC=1TO10:
NEXT:NEXT:FORA=1TO1000:NEXT:RUN
5-12 4-11 FRA FORRIGE PROGRAM
```

- 1 $A=54372$, lydstyrke, attack-deacy for første stemme, sustain-release for første stemme, attack-decay for anden stemme, sustain-release for anden stemme. Læg mærke til, at du går fra en stemme til en anden ved at lægge 7 til adressen (lydstyrke er dog fælles). Attack-decay for tredje og sidste stemme ville således blive defineret i $A+19$.
- 2 Læs tone og varighed, definer anden stemme som første stemmes værdi ganget med 2. Dette vil altid hæve en tone en oktav. Nøjagtige toneværdier i appendiks. Fordel begge toneværdier.
- 3 Spil tone, vælg bølgeformer, hold toner.
- 4 Sluk toner, pause, om igen. Spil melodien igen.

Det er altså ikke nogen rigtig anden stemme, men blot førstestemmen ført en oktav op. Andenstemmen får bølgeform 33, mens førstestemmen får 17, hvilket giver en slags fløjte-akkompagnement. Prøv at udskifte $E=C*2$ i 2 med $E=C*4$ eller $E=INT(C/2)$ eller byt om på stemmerne eller skift værdierne ud i attack etc.

Du kan naturligvis også lave en rigtig anden stemme ved at kopiere bassen fra noder. Endelig bliver dine spil selvfølgelig helt anderledes, ved at musik og lydeffekter kommer ind i dem.

Der er et helt lille musikinstrument indbygget i COMMODE 64, der kan udnyttes til det utrolige. Dette har imidlertid mere med musikkundskab end egentlig programmering at gøre, hvorfor jeg overlader det til dem blandt læserne, der har de nødvendige forudsætninger.

LEKTION 38

JOYSTICK

Sæt joystick (hvis du har en) til port 1 og kør

```
1 PRINTPEEK(56321):RUN
```

Den skriver 255, til vi vrider på joysticken. Lad os prøve at finde ræsonen i det og samtidig øve os i at undersøge systemvariabler, kernepunktet, hvis du, når du har lagt denne bog fra dig, vil udforske din maskine yderligere. Til en begyndelse foreslår jeg

```
1 A=PEEK(56321):FORB=7TO0STEP-1:IFA>=2p  
  BTHENPRINT"1";:A=A-2pB:GOTO3  
2 PRINT"0";  
3 NEXT:PRINT:RUN
```

Det er nu meget let at se, at bit 7-5 er fuldstændig upåvirkede af joystickens position. For de sidste 5 bits gælder

bit 4	bit 3	bit 2	bit 1	bit 0
fyr	øst	vest	syd	nord

eksempelvis slukkes altså bit 2, når sticken føres mod vest. Tilbage står nu at udtrykke dette praktisk i et basic-program (en nord-østlig bevægelse bliver selvfølgelig til 10110).

Vi kan starte med at droppe de overflødige bits og trække 245 fra:

1 PRINTPEEK(56321)-245:RUN

Vi får så værdierne

```
0  SØ
1  NØ
2  Ø
3  -
4  SV
5  NV
6  V
7  -
8  S
9  N
10 STILSTAND
```

Herudfra kan du nemt lave et program, f.eks.

```
1 POKE53280,1:POKE53281,1:A=10:B=20:PRINT"h"
2 POKEA*40+B+983,160:POKEA*40+B+55255,0:C=
  PEEK(56321)-247
3 A=A+(C=7)-(C=6):B=B+(C=4)-(C=0):A=A+(A=26)-
  (A=0):B=B+(B=41)-(B=0):GOTO2
```

Skal vi bruge affyringsknappen, kan vi sige noget med

```
IFC=-8THEN...
```

altså når vi vel at mærke som i ovenstående eksempel vælger at trække 247 fra. Eller måske er

```
IFC < 0THEN...
```

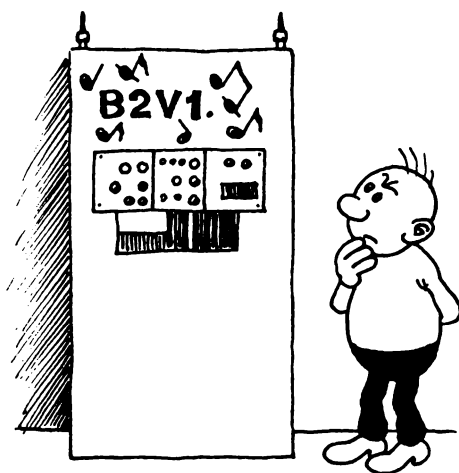
at foretrække, eftersom vi så også kan vride på sticken,

mens vi fyrer (hvis man tilgiver den ambivalente formulering).

En joystick i port 2 vil tilsvarende give udslag i 56320.

Den centrale pointe er imidlertid, at du som i det ovenstående selv kan gå på opdagelse i systemvariablerne og således gennem, hvad du har lært, kan nå ud over, hvad du har lært, hvilket filosofisk set er kendetegnet på al sand lærdom. Her vil du finde en betydelig hjælp i **COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE**, som din forhandler let kan skaffe dig. Den er ikke nogen lærebog, men med baggrund i den bog, du lige har læst, skulle du kunne få glæde af de mange tabeller.

Der er med andre ord lagt op til, at du selv vover dig videre ind i datajungen . . .



LEKTION 39

DISKETTE

I lektion 19 lærte vi at gemme programmer på kassettebånd. Det kan også gøres på en anden måde. Lad os antage, at du er i besiddelse af en diskettestation og se på, hvordan sådan en virker.

Vi kan starte med at komme en frisk diskette i stationen (husk at lukke lågen). Du har nu en »ekstra« hukommelse at arbejde med, som er en hel del mere elastisk end et kassetebånd. Du kan oplagre programmer i den, men du kan også simpelt hen fylde den med data, altså f.eks. et kapitel af en roman. Så skal du bruge ordren

PRINT #

PRINT ville skrive dine data ud på skærmen, PRINT # skriver dem ind på disketten. De kan imidlertid ikke hænge frit i luften eller rasle rundt på må og få på disketten. Du må åbne en såkaldt fil til dem, endnu en slags skuffe, men selvfølgelig en hel del mere rummelig end lektion 1's variabel. Indskrivning af data på diskette foregår således i tre trin: Åben fil, indskriv data, og luk fil, eller OPEN, PRINT # og CLOSE.

En fil kan nummereres med et tal fra 1 til 127, men du skal også kunne nå filerne fra tastaturet, og til det brug har du kanalerne 0-15, dvs. 0 og 1 bruger maskinen sådan set selv. Tilbage er 2-14 til data-overførsel og 15, som er kommando-kanalen, hvorigennem du sender diskettestationen de nødvendige kommandoer. Sådanne kommandoer er

blandt andet nødvendige, når du som nu starter på en helt ny diskette. Maskinen skal sætte visse kendemærker på den, som den senere går efter, når den bruger den. Hertil bruges ordren NEW (som selvfølgelig også kan bruges til at »slette« en diskette). Lad os forsøge.

Første trin er at åbne vores kommandokanal. Det gør vi med

OPEN15,8,15

hvilket betyder så meget som: Åben fil 15, diskteststation, kanal 15 (kommando-kanal). Som du vil forstå betyder 8 diskette. LOAD»PROGRAM«,8 tager et program ind fra diskette. LOAD henter, som vi ved (uden noget tal angivet), et program fra båndstationen.

Nu kan vi sende vores kommando:

PRINT # 15,"NEW0:TEST,AA"

Som du ser, danner kommandoen en streng, som vi »printer ind« i fil nummer 15. Denne kommando består af fire oplysninger:

NEW NY (slet) disk

0 Diskteststation 1 (du kan bruge to)

TEST Navn på hele disken

AA (Altid 2) kendingsbogstaver, der hjælper til at identificere disketten

Vi kan nu prøve at SAVE et program på diskette, f.eks. DEN RØDE BARON fra lektion 35. Skriv simpelt hen

SAVE"DEN RØDE BARON",8

Maskinen er færdig, næsten før den er begyndt. Også på diskette kan du verificere. Skriv

VERIFY"DEN RØDE BARON",8

Slet nu programmet i maskinen med NEW og tag det ind igen med

LOAD"DEN RØDE BARON",8

Alt som på kassette. Men prøv nu

LOAD"\$",8

fulgt af

LIST

og du har en indholdsfortegnelse, der først giver dig navn og kendingsbogstaver på disketten og derefter lister dens indhold, altså i vores tilfælde

5 "DEN RØDE BARON" PRG

altså PRoGrammet »DEN RØDE BARON«, der optager 5 »blokke« af diskettens kapacitet, og så er i øvrigt stadig 659 blokke fri, vi har altså end ikke brugt en enkelt procent af den. Prøv at lægge disketten væk og slukke for hele herligheden. Tænd igen, skriv

LOAD"\$",8

og

LOAD"DEN RØDE BARON",8

Det hele er der endnu. Lad os prøve igen, med f.eks. »SPRITE«, generatoren fra lektion 33. Dette program kan være på 2 blokke, og vi har 657 i reserve. Lad os se, hvad mere vi kan lave. Måske bliver vi trætte af første verdenskrig. Så kan vi faktisk blive af med programmet, uden at det går ud over SPRITE. Nu skal vi igen snakke med stationen, så vi lægger ud med at åbne vores kommandokanal

```
OPEN15,8,15
```

Nu bruger vi kommandoen SCRATCH og siger

```
PRINT # 15,"SCRATCH0:DEN RØDE BARON"
```

```
Tag en LOAD"$",8
```

og du vil se, at vi har fået vores 5 blokke igen.

Vi finder måske også ud af, at SPRITE er et dårligt navn, og SPRITEGENERATOR ville være mere udtømmende. Så kan vi skifte med RENAME:

```
PRINT # 15,"RENAME0:SPRITEGENERATOR=SPRITE"
```

Til sidst checker vi indholdsfortegnelsen og kontrollerer, at alt er sket fyldest. Vi kan sågar lave en kopi af et program på disketten under et andet navn:

```
PRINT # 15,"COPY0:TEGN=0:SPRITEGENERATOR"
```

Et program kan jo smutte, også på en diskette, og så er en reserve god at have. Eller lad os rette i TEGN, så der kommer et andet program ud af det:

```
LOAD"TEGN",8
```

Vi kunne så f.eks. rette 2-tallet i POKE nummer 2 i linje 2 til et 6-tal, så vi får blå prikker i stedet for røde. I stedet for nu at scratche dette gamle program og loade det nye, kan vi nøjes med

SAVE" @0:TEGN",8

NEW og LOAD"TEGN",8

RUN og du har de blå prikker, men LOAD"\$",8 vil vise, at der ikke ved siden af findes et TEGN med røde prikker. Det har vi jo til gengæld i SPRITEGENERATOR.

LEKTION 40

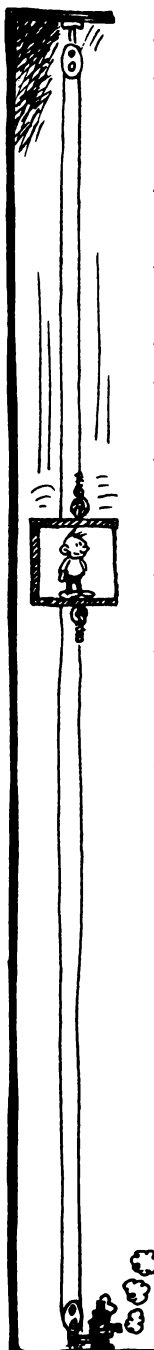
TIDSMASKINEN

```
1 POKE53280,12:POKE53281,15:PRINT"hsf7f9TIDS  
  MASKINEN"  
2 PRINT"ssDU ER EN GENIAL OPFINDER, UHEL  
  DIGVIS ER"  
3 PRINT"DIN BROR GEORG IKKE JUST KVIK.":  
  PRINT"sEN DAG STIKKER HAN AF MED DIN"  
4 PRINT"TIDSMASKINE, OG SNART EFTER MOD  
  TAGER DU":PRINT"HANS SOS."  
5 PRINT"sTAST MELLEMRUM OG DEREFTER  
  DET ANTAL AAR"  
6 PRINT"n(FULGT AF RETURN), DU SKAL SENDE  
  HAM"  
7 PRINT"FREM I TIDEN FOR AT BRINGE HAM  
  TILBAGE"  
8 PRINT" Til 1990, HVOR HAN STARTEDE."  
9 PRINT"shVIS DU IKKE SENDER HAM LANGT  
  NOK, KAN"  
10 PRINT"DU PRØVE IGEN, MEN DU KAN IKKE  
  HENTE HAM"  
11 PRINT"nTILBAGE FRA FREMTIDEN!":PRINT"ss  
  HELD OG LYKKE – TAST RETURN . . . s"  
12 DIMA$(99),B$(99),A(99):FORA=1TO99:READA$(A),  
  B$(A),A(A):NEXT  
13 FORA=0TO9:READC$(A):NEXT:POKE2040,14:F  
  ORA=896TO958:READB:POKEA,B:NEXT  
14 A=53248:POKEA+21,1:POKEA+23,1:POKEA+29,1  
15 POKEA+1029,255:POKEA+1030,255:POKEA+10  
  36,255:POKEA+1037,255:POKEA+1048,15
```

```

16 GETA$:IFA$=""THEN16
17 FORB=1TO153:PRINT"c8"INT(RND(1)*991)+1000
  ;:NEXT:PRINT"f3":FORB=1424TO1623
18 POKEB,160:POKEB+54272,6:NEXT:B=1:C=
  1:D=INT(RND(1)*99):TI$="000000"
19 A$=A$(D)+"m- SOS - JEG TALER MEDm"+B$(D)
  +"mSOMm"+C$(A(D))
20 A$=A$+"m- SOS - JEG VIL HJEM -m"
21 PRINT"hssssssssssss0000000000f9"LEFT$(A$,20)
22 A$=RIGHT$(A$,LEN(A$)-1)+LEFT$(A$,1):GETB$:
  IFB$="m"THEN31
23 POKEA,E:POKEA+1,F:POKEA+1028,17:POKEA+
  1025,INT(E/2):POKEA+1028,16
24 POKEA+1035,33:POKEA+1032,255-F:POKEA+
  1035,32
25 E=E+B:F=F+C:IFE=0ORE=255ORRND(1)<.01
  THENB=-B:POKEA+39,INT(RND(1)*16)
26 IFF=0ORF=255ORRND(1)<.01THENC=-C:POK
  EA+39,INT(RND(1)*16)
27 IFRND(1)<.001THENPOKEA+32,INT(RND(1)*16)
28 G=INT(RND(1)*153):IFG>59ANDG<94THEN28
29 G=G*6+55337:H=INT(RND(1)*16):FORI=GTOG
  +3:POKEI,H:NEXT
30 GOTO21
31 PRINT"s0000000000f9mmmmmmmmmmmmmm
  mmmmmmm":INPUT"nn0000000000f9";G:J=J
  +1
32 G=INT(G/10):D=D+G
33 IFD=100THENPRINT"nnf9000GEORG HJEMME
  IGEN -";B=INT((20-J)*100-TI/100):GOTO36
34 IFD<100THEN19
35 PRINT"nnf9000GEORG HAVNEDE I"D*10+990
  "-";B=INT((120-D-J)*100-TI/100)
36 IFB<0THENB=0
37 PRINTB"vmPOINTS"

```



- 38 POKEA+1048,0:GOTO38
- 39 DATAMADJARERNE HAR SLUTTET SIG TIL
ROMERKIRKEN – SLAG VED SVOLDER,VLADI
MIR,5
- 40 DATADANSKERHERREDØMMET I ENGLAND,
VLADIMIR,6
- 41 DATANORDINDIEN ER BLEVET EROBRET AF
MAHMUD AF GHANZI,KNUD DEN STORE,1
- 42 DATASLAGET VED STIKLESTAD,ANSELM,
- 43 DATAHEDEBY ER BLEVET UDSLETTET,AN
SELM,1
- 44 DATAPAVEN OG PATRIARKEN I KONSTANTI
NOPEL HAR LYST HINANDEN I BAND, AN
SELM,2
- 45 DATANI BISPEDØMMER I DANMARK,AN
SELM,3
- 46 DATANORMANNERNE HAR EROBRET SYD
ITALIEN – SIMONI STRAFFES MED BAND,AN
SELM,4
- 47 DATAKNUD DEN HELLIGE ER BLEVET MYR
DET,PIERRE ABAILARD,1
- 48 DATAKIRKEMØDE I CLERMONT – ANTIOKIAS
EROBRING,PIERRE ABAILARD,2
- 49 DATADOMKIRKER I LUND – VIBORG – RIBE,
IDRISI,1
- 50 DATASIGURDJORSALFARS KORSTOG –
BØHMENS FYRSTE TYSK KURFYRSTE,IDRISI,2
- 51 DATAKONKORDATET I WORMS,IDRISI,3
- 52 DATASICILIEN OG SYDITALIEN ER BLEVET
FORENET TIL EET RIGE,IDRISI,4
- 53 DATAHERIDSVAD KLOSTER,IDRISI,5
- 54 DATARIGSDAGEN I MERSEBURG – FREDERIK
BARBAROSSA KEJSERKRONET I ROM,IDRISI,4
- 55 DATAABSALONS BORG VED HAVN – ARKO
NAS EROBRING,FREDERIK BARBAROSSA,1

- 56 DATATHOMAS BECKET ER BLEVET MYR
DET,FREDERIK BARBAROSSA,2
- 57 DATARIGSDAGEN I MAINZ – BAJERN OVER
DRAGES TIL WITTELSBACHERNE,SALADIN,1
- 58 DATAFREDERIK BARBAROSSA DRUKNET I
LILLEASIEN,SALADIN,2
- 59 DATAHOLSTEN ER BLEVET EROBRET AF
DANSKERNE,JOHAN UDEN LAND,1
- 60 DATASLAGET VED BOUVINES – SYNODEN I
LATERANET,DJENGIS KHAN,1
- 61 DATADOMINIKANERORDENEN – FRANCISKA
NERORDENEN,DJENGIS KHAN,2
- 62 DATADEN PAVELIGE INKVISITION – TIGGER
MUNKE TIL DANMARK,RUDOLF AF HABS
BURG,2
- 63 DATAKONGSEMNEKRIGENE OPHØRT – SLA
GET VED LIEGNITZ,THOMAS AQUINAS,2
- 64 DATAGRANADA SIDSTE MAURISKE BESIDDEL
SE I SPANIEN,KARL AF ANJOU,3
- 65 DATAISLAND OG GRØNLAND UNDER NOR
GES KONGE,KUBLAI KHAN,
- 66 DATAMARCO POLOS REJSE TIL KINA – KIR
KEMØDET I LYON,DANTE ALIGHIERI,1
- 67 DATASICILIEN UNDER ARAGONIEN – MOR
DET I FINDERUP,MARCO POLO,3
- 68 DATAAKKON ER BLEVET EROBRET AF MA
MELUKKERNE – PARLAMENTET I IRLAND,GI
OTTO,2
- 69 DATAVISCONTIERNE MAGTEN I MILANO –
POLEN SAMLES,GIOTTO,3
- 70 DATASLAGET VED BANNOCKBURN – EDS
FORBUNDET I BRUNNEN,FRANCESCO PET
RARCA,1
- 71 DATADANTE ALIGHIERI DØR,DANTE ALIGHI
ERI,8

- 72 DATAAZTEKERNE BYGGER TENOCHTITLAN,
JOHN WYCLIFFE,1
- 73 DATASLAGET VED CRECY – ENGELSK SØSEJR
VED SLUIS,MUHAMED TUGHLUK,2
- 74 DATABIRGITTINERORDENEN – DEN SVENSK
LANDSLOV,GEOFFREY CHAUCER,1
- 75 DATADANEHOF I KALUNDBORG,GEOFFREY
CHAUCER,2
- 76 DATASTRALSUNDFREDEN,JOHAN HUS,1
- 77 DATADEN ENGELSKE BONDEOPSTAND ER
BLEVET KUET,JOHAN HUS,2
- 78 DATAOSMANNERNE HAR EROBRET THESSA
LIEN OG BULGARIEN,JOHAN HUS,3
- 79 DATASLAGET VED ANKARA,JAN VAN EYCK,1
- 80 DATADEN TYSKE ORDENS NEDERLAG VED
TANNENBERG,HENRIK SØFAREREN,2
- 81 DATAFORLIGET I TORYES – SUNDTOLDEN,JO
HAN GUTENBERG,2
- 82 DATAKONCILET I BASEL – COSIMO MEDICI
MAGTEN I FIRENZE,NICOLO DE CONTI,2
- 83 DATAINKARIGET – DET NORDJYSKE BONDE
OPRØR,FRANCISCO JIMENEZ,1
- 84 DATAROSEKRIGENE I ENGLAND,DONATO
BRAMANTE,1
- 85 DATAOVERENSKOMSTEN I RIBE – FREDEN I
THORN,HENRIK SØFAREREN,7
- 86 DATAHOLSTEN HERTUGDØMME – UNIVERSI
TET I UPPSALA,GIOVANNI CABOTTO,2
- 87 DATAMONGOLERNE ER BLEVET FORDREVET
FRA RUSLAND,BARTHOLOMEU DIAZ,3
- 88 DATAFØRSTE DELING I SLEVSVIG OG HOL
STEN,CHRISTOPHER COLUMBUS,4
- 89 DATADITMARSKERTOGET – RIGSDAG I AUGS
BURG,AMERIGO VESPUCCI,5

- 90 DATABALBOA HAR OPDAGET DET STORE
OG STILLE OCEAN,LEONARDO DA VINCI,6
- 91 DATALUTHERS BANDLYSNING – DET STOCK
HOLMSKE BLODBAD,VASCO DA GAMA,6
- 92 DATARIGSDAG I AUGSBURG,ERASMUS AF
ROTTERDAM,7
- 93 DATAXAVIER TIL INDIEN – DEN PAVELIGE
INKVISATION FORNYET,MARTIN LUTHER,6
- 94 DATACOLLEGIUM ROMANUM – COLLEGIUM
GERMANICUM,IGNATIUS LOYOLA,6
- 95 DATAREFORMATION I SKOTLAND – JANUAR
EDIKTET I FRANKRIG,JEAN CALVIN,6
- 96 DATAFREDEN I STETTIN – SØSLAGET VED
LEPANTO,GERHARD MERCATOR,6
- 97 DATAPORTUGAL ER BLEVET FORENET MED
SPANIEN,DEN HELLIGE TERESA,7
- 98 DATAKIRKEMØDET I UPPSALA – BANCO DE
RIALTO,MIGUEL DE CERVANTES,5
- 99 DATADET ENGELSKE OSTINDISKE HANDELS
SELSKAB,MATTEO RICCI,5
- 100 DATAKALMARKRIGEN – FREDEN I STOLBO
VA,WILLIAM SHAKESPEARE,5
- 101 DATASLAGET VED DET HVIDE BJERG – PLY
MOUTH,GALILEO GALILEI,6
- 102 DATAGUSTAV ADOLF LANDET I POMMERN,
SAMUEL DE CHAMPLAIN,3
- 103 DATAUNIONEN MELLEM PORTUGAL OG SPA
NIEN OPLØSES,GALILEO GALILEI,8
- 104 DATANAVIGATIONSAKTEN I ENGLAND,WIL
LIAM HARVEY,8
- 105 DATAFREDEN I KØBENHAVN – BAMBORARI
GET VED ØVRE NIGER,GIOVANNI BERNINI,7
- 106 DATADEN NYE DANSKE ADEL,JOHN MIL
TON,7

- 107 DATASTRASSBURG TIL FRANKRIG,JEAN BAPTISTE COLBERT,7
- 108 DATASLAGET VED BOYNEFLODEN – PAPINS DAMPMASKINE I FRANKRIG,JOHN LOCKE,6
- 109 DATADEN GREGORIANSKE KALENDER ER BLEVET INDFØRT I DANMARK,OLE RØMER,6
- 110 DATASLAGET VED HELSINGBORG – WHIG GERNE STYRTET – PESTEN I KØBENHAVN, ANNA,1
- 111 DATAKONGERIET SARDINIEN – FREDEN I FREDERIKSBORG,ISAAC NEWTON,8
- 112 DATADEN POLSKE TRONFØLGERKRIG – FREDEN I WIEN,DANIEL DEFOE,7
- 113 DATADEN ØSTRIGSKE ARVEFØLGEKRIG – VIDENSKABERNES SELSKAB,VITUS BERING,6
- 114 DATASEMINARIET I TØNDER,JOHANN SEBASTIAN BACH,7
- 115 DATAFREDEN I HUBERTSBURG – FREDEN I PARIS,JACQUES VAUCANSON,6
- 116 DATASTRUENSEES REGERING I DANMARK,JEAN JACQUES ROUSSEAU,6
- 117 DATASØNDAGSSKOLER BEGYNDT I ENGELAND,DENIS DIDEROT,7
- 118 DATATRYKKEFRIHEDEN ER BLEVET SIKRET I DANMARK,GEORGE WASHINGTON,6
- 119 DATASLAG VED MARENGO – REALUNION MED IRLAND,FRIEDRICH VON SCHILLER,5
- 120 DATAHOLLAND ER BLEVET INDLEMMET I FRANKRIG,WARREN HASTINGS,8
- 121 DATAREVOLUTIONEN I SPANIEN – REVOLUTIONEN I PORTUGAL,EDMUND CARTWRIGHT,8
- 122 DATAJULIREVOLUTIONEN,JOHANN WOLFGANG VON GOETHE,9

- 123 DATAKRIGEN MELLEM ENGLAND OG KINA,
GEORGE STEPHENSON,6
- 124 DATAFORFATNINGEN I PREUSSEN,HERTUGEN
AF WELLINGTON,9
- 125 DATAHANDELSTRAKTATEN MELLEM FRAN
KRIG OG ENGLAND,ABRAHAM LINCOLN,6
- 126 DATATELEFONERING – SLAGET VED SEDAN
– DET FORENEDE VENSTRE DANNES,SHA
MYL,8
- 127 DATADEN TVUNGNE SKOLEGANG I ENGLAND
–FRANKRIG HAR TAGET TUNIS,KARL MARX,7
- 128 DATABISMARCK AFSKEDIGET – TYSKLAND
HAR KØBT HELGOLAND,OTTO VON BIS
MARCK,8
- 129 DATABOXEROPSTANDEN I KINA – DEN ØKO
NOMISKE VERDENSKRISE,GUISEPPE VERDI,9
- 130 DATAPORTUGAL REPUBLIK – DEN FASTE
VOLDGIFTSRET I DANMARK,JOSEPH LISTER,9
- 131 DATANORDSLESVIG DANSK EFTER FOLKEAF
STEMNING,NICOLAI LENIN,5
- 132 DATABRIANDS FORSLAG OM EUROPAS FOR
ENEDE STATER,SIGMUND FREUD,8
- 133 DATACHURCHILL HAR DANNET REGERING I
ENGLAND,GEORGE BERNHARD SHAW,9
- 134 DATASCHUMANPLANEN ER BLEVET OFFENT
LIGGJORT,ALBERT EINSTEIN,8
- 135 DATADET FRANSKE OPRØR I ALGIER IMOD
DE GAULLE,WINSTON CHURCHILL,9
- 136 DATAKINAS FØRSTE SATELLIT – KRISEN I
JORDAN,RICHARD MILHOUS NIXON,6
- 137 DATAKRIG I MELLEMØSTEN – ARBEJDSLØS
HED I DANMARK,ELSEBETH,3
- 138 DATALILLE,BARN,PURUNG,UNG,YNGRE,MO
DENT MENNESKE,MIDALDRENDE,GAMMEL
- 139 DATAMEGET GAMMEL,URGAMMEL

140 DATA4,3,128,127,199,192,21,15,224,63,152,48,4,63,
248,4,96,8,4,248,8,5,184,8
141 DATA7,178,8,7,191,8,7,176,136,15,190,120,21,190,
72,53,230,72,78,103,72,251
142 DATA255,248,78,88,52,53,143,226,53,7,193,127,255,
255,132,1,1
143 REMA=53248:B=XFLAG:C=YFLAG:D=ANNO:E
=X:F=Y:G=INPUT:H=FARVE:I=KONTROLVARI
ABEL
144 REMJ=OMGANG
145 REMDIN TUR – GOD FORNØJELSE...!

APPENDIKS

KODER

fA Tallet A med CTRL holdt nede
A Tasten A med SHIFT holdt nede
cA Tasten A med COMMODORE holdt nede
h Home
h Clr
m Mellemlrum
n Markør op
s Markør ned
ø Markør til højre
v Markør til venstre
p Potens
i Insert

I de første tre eksempler står A for vilkårlig tast eller tal. Et hjerte er således S, tasten S med SHIFT holdt nede.

Fejlkoder kan du finde i indeks.

Visse af de reserverede ord i BASIC kan forkortes til det første bogstav efterfulgt af det andet bogstav med SHIFT holdt nede. Dette er brugt enkelte steder af pladshensyn.

SKÆRMKODER FOR KARAKTERSÆTTENE

Tabellen viser hvilke værdier man skal POKE ind i skærmhukommelsen for at få de pågældende karakterer frem på skærmen.

1. sæt	2. sæt		1. sæt	2. sæt		1. sæt	2. sæt
@		0	U	u	21	*	42
A	a	1	V	v	22	+	43
B	b	2	W	w	23	,	44
C	c	3	X	x	24	—	45
D	d	4	Y	y	25		46
E	e	5	Z	z	26	/	47
F	f	6	[		27	Ø	48
G	g	7	£		28	1	49
H	h	8	]		29	2	50
I	i	9	↑		30	3	51
J	j	10	←		31	4	52
K	k	11	SPACE		32	5	53
L	l	12	!		33	6	54
M	m	13	“		34	7	55
N	n	14	#		35	8	56
O	o	15	\$		36	9	57
P	p	16	%		37		58
Q	q	17	&		38	;	59
R	r	18	'		39	<	60
S	s	19	(		40	=	61
T	t	20	)		41	>	62

1. sæt 2. sæt

?	63
	64
A	65
	66
	67
	68
	69
	70
	71
	72
	73
	74
	75
	76
	77
	78
	79
	80
	81
	82
S	83

1. sæt 2. sæt

	T	84
	U	85
	V	86
	W	87
	X	88
	Y	89
	Z	90
		91
		92
		93
		94
		95
		96
		97
		98
		99
		100
		101
		102
		103
		104
		105

1. sæt 2. sæt

	106
	107
	108
	109
	110
	111
	112
	113
	114
	115
	116
	117
	118
	119
	120
	121
	122
	123
	124
	125
	126
	127



ASCII-KODER

Tabellen viser sammenhængen mellem karakterer og kodeværdier ved brug af funktionerne ASC(A\$) og CHR\$(A), hvor A\$ er en karakter (husk anførelsestegn) og A er et tal (den tilsvarende kodeværdi).

Karakter	Kode	Karakter	Kode	Karakter	Kode	Karakter	Kode
	0		16	SPACE	32	Ø	48
	1	CRSR ↓	17	!	33	1	49
	2	RVS ON	18	"	34	2	50
	3	CLR HOME	19	#	35	3	51
	4	INST DEL.	20	\$	36	4	52
WHT	5		21	%	37	5	53
	6		22	&	38	6	54
	7		23		39	7	55
	8		24	(	40	8	56
	9		25	)	41	9	57
	10		26	.	42		58
	11		27	+	43	;	59
	12	RED	28	,	44	<	60
RETURN	13	CRSR ↓	29	—	45	=	61
SWITCH TO LOWER CASE	14	GRN	30		46	>	62
	15	BLU	31	/	47	?	63

Karakter	Kode	Karakter	Kode	Karakter	Kode	Karakter	Kode
@	64	U	85		106		127
A	65	V	86		107		128
B	66	W	87		108	Orange	129
C	67	X	88		109		130
D	68	Y	89		110		131
E	69	Z	90		111		132
F	70	[	91		112	f1	133
G	71	£	92		113	f3	134
H	72	]	93		114	f5	135
I	73	↑	94		115	f7	136
J	74	←	95		111	f2	137
K	75		96		117	f4	138
I	76		97		118	f6	139
M	77		98		119	f8	140
N	78		99		120	SHIFT	141
O	79		100		121	RETURN	142
P	80		101		122	SWITCH TO UPPER CASE	143
Q	81		102		123	BLK	144
R	82		103		124	CRSR	145
S	83		104		125	RVS OFF	146
T	84		105		126	CLR HOME	147

Karakter	Kode	Karakter	Kode	Karakter	Kode	Karakter	Kode
	148		159		170		181
Brun	149		160		171		182
Lyserød	150		161		172		183
Grå 1	151		162		173		184
Grå 2	152		163		174		185
Lysegrøn	153		164		175		186
Lyseblå	154		165		176		187
Grå 3	155		166		177		188
	156		167		178		189
	157		168		179		190
	158		169		180		191

TONEVÆRDIER

Tone	Værdi
C-0	268
C#-0	284
D-0	301
D#-0	318
E-0	337
F-0	358
F#-0	379
G-0	401
G#-0	425
A-0	451
A#-0	477
B-0	506
C-1	536
C#-1	568
D-1	602
D#-1	637
E-1	675
F-1	716
F#-1	758
G-1	803
G#-1	851
A-1	902
A#-1	955
B-1	1012
C-2	1072
C#-2	1136
D-2	1204
D#-2	1275
E-2	1351
F-2	1432
F#-2	1517

Tone	Værdi
G-2	1607
G#-2	1703
A-2	1804
A#-2	1911
B-2	2025
C-3	2145
C#-3	2273
D-3	2408
D#-3	2551
E-3	2703
F-3	2864
F#-3	3034
G-3	3215
G#-3	3406
A-3	3608
A#-3	3823
B-3	4050
C-4	4291
C#-4	4547
D-4	4817
D#-4	5103
E-4	5407
F-4	5728
F#-4	6069
G-4	6430
G#-4	6812
A-4	7217
A#-4	7647
B-4	8101
C-5	8583
C#-5	9094

Tone	Værdi
D-5	9634
D#-5	10207
E-5	10814
F-5	11457
F#-5	12139
G-5	12860
G#-5	13625
A-5	14435
A#-5	15294
B-5	16203
C-6	17167
C#-6	18188
D-6	19269
D#-6	20415
E-6	21629
F-6	22915
F#-6	24278
G-6	25721
G#-6	27251
A-6	28871
A#-6	30588
B-6	32407
C-7	34334
C#-7	36376
D-7	38539
D#-7	40830
E-7	43258
F-7	45830
F#-7	48556
G-7	51443
G#-7	54502
A-7	57743
A#-7	61176
B-7	64814

SLAG

```
1 GOSUB43:POKE53281,0:PRINT"h":FORA=0TO3:
  FORB=0TO7
2 C=5-(RND(1)<.1):IFRND(1)<.1THENC=2
3 IF(B<2ORB>7-J)ANDC=6THEN2
4 POKEA*200+B*5+1024,A+176:POKEA*200+B*
  5+1025,B+176:IFB>1THEN7
5 D=A*200+B*5+1065:POKED,57:POKED+1,77:P
  OKED+40,109:POKED+41,78
6 D=D+54272:POKED,2:POKED+1,2:POKED+40,2
  :POKED+41,2
7 IFB<8-JTHEN10
8 D=A*200+B*5+1065:POKED,57:POKED+1,110:
  POKED+40,77:POKED+41,125
9 D=D+54272:POKED,6:POKED+1,6:POKED+40,6:
  POKED+41,6
10 FORD=0TO3:FORE=0TO3
11 IFD=0ANDE>1ORD=3ORDANDE=0ORE=3THEN
  POKEA*200+B*5+D*40+E+1024,160
12 IFD=0ORD=3ORE=0ORE=3THENPOKEA*200+B*
  5+D*40+E+55296,C
13 NEXTE,D,B,A:K=1
14 PRINT"sssssssssssssssf2"
15 INPUT"FRA";A$:IFLEN(A$)<>2THENPRINT"nn"
  :GOTO15
16 A=ASC(LEFT$(A$,1))-48:B=ASC(RIGHT$(A$,1))-48
17 IFA<0ORA>3ORB<0ORB>7THENPRINT"nn"
  :GOTO15
```

```

18 C=A*200+B*5+55337:IFPEEK(C)<>2THENPRINT
   "nn":GOTO15
19 PRINT"nn"
20 INPUT"TIL";A$:IFLEN(A$)<>2THENPRINT"nn"
   :GOTO15
21 D=ASC(LEFT$(A$,1))-48:E=ASC(RIGHT$(A$,1))-48
22 IFD<0ORD>3ORE<0ORE>7THENPRINT"nn"
   :GOTO15
23 IFABS(A-D)>1ORABS(B-E)>1THENPRINT"nn":
   GOTO15
24 F=D*200+E*5+55337:IFPEEK(F)<>0ORPEEK(F-
   41)=6THENPRINT"nn":GOTO15
25 GOSUB56:POKEF,2:POKEF+1,2:POKEF+40,2:POK
   EF+41,2
26 G=INT(RND(1)*4):H=INT(RND(1)*8):FORA=GTO3
   :FORB=HTO7:C=A*200+B*5+55337
27 IFPEEK(C)=6THENONINT(RND(1)*10)+1GOTO29,
   29,29,29,29,29,29,30,31
28 NEXTB,A:GOTO26
29 IFB>0THENF=C-5:IFPEEK(F)=0THEN38
30 IFA>0ANDB>0THENF=C-205:IFPEEK(F)=0THEN
   38
31 IFA<3ANDB>0THENF=C+195:IFPEEK(F)=0THEN
   38
32 IFA>0THENF=C-200:IFPEEK(F)=0THEN38
33 IFA<3THENF=C+200:IFPEEK(F)=0THEN38
34 IFA>0ANDB<7THENF=C-195:IFPEEK(F)=0THEN
   38
35 IFA<3ANDB<7THENF=C+205:IFPEEK(F)=0THEN
   38
36 IFB<7THENF=C+5:IFPEEK(F)=0THEN38
37 GOTO28
38 IFPEEK(F-41)=6THEN28
39 H=F-55337:D=INT(H/200):E=(H-INT(H/200)*200)/5

```

```

40 GOSUB56:POKEF,6:POKEF+1,6:POKEF+40,6:
   POKEF+41,6
41 H=2:I=6:GOSUB62:H=6:I=2:GOSUB62:L=L+1:IF
   L=24THENL=0:K=K+1
42 PRINT"sDATO:"K:PRINT"KLOKKEN:"L"vm":PRIN
   T"nnnnn":GOTO15
43 POKE53280,11:POKE53281,11
44 PRINT"hƒ9c5SLAG":PRINT"sSLAG i LANDSKAB
   MED (RØDE) LANDSBYER OG"
45 PRINT"SØER. DU ER RØD. RYK VED AT TASTE"
46 PRINT"FELTETS NUMMER. DU KAN RYKKE ET
   FELT I"
47 PRINT"EN AF OTTE RETNINGER. TALLET I KO
   LONNENS"
48 PRINT"nØVERSTE VENSTRE HJØRNE ER DENS
   STYRKE."
49 PRINT"sDU ER BEDST BESKYTTET I BEBYGGEL
   SE."
50 PRINT"sEFTER SOM DU BLIVER DYGTIGERE,
   KAN DU"
51 PRINT"SKIFTE NIVEAU."
52 PRINT"sGOD FORNØJELSE!s"
53 INPUT"NIVEAU(1-5)";A$:IFLEN(A$)< > 1THEN
   PRINT"nn":GOTO53
54 IFASC(A$)< 49ORASC(A$)> 53THENPRINT"nn":
   GOTO 53
55 J=ASC(A$)-48:RETURN
56 G=F-54272:H=PEEK(C-54272)
57 POKEC,0:POKEC+1,0:POKEC+40,0:POKEC+41,0
   :POKEG,H
58 IFA < DTHENPOKEG+1,110:POKEG+40,77:POKEG
   +41,78:RETURN
59 IFA > DTHENPOKEG+1,77:POKEG+40,109:POKEG
   +41,125:RETURN

```

```

60 IFB < ETTHENPOKEG+1,77:POKEG+40,109:POKEG
   +41,78:RETURN
61 IFB > ETTHENPOKEG+1,110:POKEG+40,77:POKEG
   +41,125:RETURN
62 FORA=0TO3:FORB=0TO7:C=A*200+B*5+55337
   :IFPEEK(C)=HTHEN64
63 NEXTB,A:RETURN
64 D=C-54272:IFA > 0ANDPEEK(D+1)=77ANDPEEK
   (D+41)=125THENE=D-200:GOTO69
65 IFB < 7ANDPEEK(D+1)=77ANDPEEK(D+41)=78THE
   NE=D+5:GOTO69
66 IFA < 3ANDPEEK(D+40)=77ANDPEEK(D+41)=78
   THENE=D+200:GOTO69
67 IFB > 0ANDPEEK(D+40)=77ANDPEEK(D+41)=125
   THENE=D-5:GOTO69
68 GOTO63
69 F=E+54272:IFPEEK(F) < > ITHEN63
70 G=PEEK(E)
71 IFH=2ANDRND(1) < .5ORH=6THENG=G-2:IFPEEK
   (F-41)=2THENG=G+1
72 IFG < 48THENPOKEF,0:POKEF+1,0:POKEF+40,0
   :POKEF+41,0
73 POKEE,G:GOTO63
74 REMET SPIL, JEG SELV HAR MEGET FORNØ
   JELSE AF, I GRUNDPRINCIP
75 REMUDBYG DET SELV - OG LAV ET, DER MO
   RER DIG ...!

```

REGISTER

ABS 93
ADRESSE 111
AFFYRINGSKNAP 189
AKSE 165
ALFABETISERING 83
AND 57
ANNUITETER 38
ARGUMENT 48
ASC 88
ASCII 88
ATN 93
ATTACK 182
BAD SUBSCRIPT 77
BAS 187
BASIC 13
BETINGET UDTRYK 132
BEVÆGELSE 120
BINARY DIGIT 110
BINARY SYSTEM 110
BINÆRkode 111
BIT 110
BITLOGIK 142
BLOK 193
BREAK 28
BRUGSANVISNING 7
BUETANGENS 93
BYTE 110
BØLGEFORM 184
CANT CONTINUE 56
CENTRAL PROCESSING UNIT 112
CHR\$ 88
CLEAR 21
CLEAR SCREEN 23
CLOSE 191
CLR 21
COMMODORE 64 7
COMMODORE 64 BASIC V2 13
CONT 36
CONTINUE 36
CONTROL 22
CONWAYS REGLER 79
COPY 194
COS 93
COSINUS 93
CPU 112
CRASH 136
CRSR 23
CTRL 22
CURSOR 23
DATA 83
DATABASE 82
DECAY 182
DEFFN 90
DEFINERBAR FUNKTION 90
DEL 18
DELETE 18
DELSTRENG 71
DESTINATIONSTAL 90
DIALEKT 13
DIM 77
DIMENSIONERING 78
DIREKTE ORDRE 68
DISKETTE 191
DISKETTESTATION 191
DIVISION BY ZERO 94
E 36
END 37
ERROR 17
EXP 93
EXTRA IGNORED 42
FAKULTET 68
FALSK 130
FARVEKODE 115
FEJLkode 207

FEJLMEDDELELSE 55
FIL 191
FIRKANT 184
FLAG 122
FOR 65
FORMULA TOO COMPLEX 94
FRE 51
FUNKTION 48
FUNKTIONSTAST 89
GET 88
GOSUB 76
GOTO 34
GRAFISK AFBILDNING 161
GRÆNSEVÆRDI 67
HELTALS Variabel 20
HOME 23
HUKOMMELSE 14
HVID STØJ 184
IF 44
ILLEGAL DIRECT 90
ILLEGAL QUANTITY 94
INDEKS 77
INDICERET Variabel 77
INDSÆT 27
INITIALISERING 82
INPUT 39
INSERT 27
INST 27
INT 49
JOYSTICK 188
K 111
KANAL 191
KARAKTER 15
KARAKTERGENERATOR 143
KARAKTERSÆT 135
KARTOTEK 82
KASSETTEBÅNDSTATION 97
KENDINGSBOGSTAV 192
KODE 207
KOMMANDO 191
KOMMANDOKANAL 191
KONTROL Variabel 65
KOORDINAT 162

KOORDINATSYSTEM 165
KRIGSSPIL 215
KVADRANT 165
KVADRATROD 48
LØKKE 46
LEFT\$ 69
LEN 69
LESS SIGNIFICANT BYTE 151
LET 95
LIFE 79
LINJENUMMER 24
LIST 25
LOAD 100
LOG 93
LOGARITME 93
LOGISK OPERATION 58
LSB 151
LYDBØLGE 184
LYDEFFEKT 187
LYDSTYRKE 182
LYSAVIS 72
MARGEN 55
MARKØR 15
MASKINKODE 111
MELEMMRUM 15
MID\$ 69
MORE SIGNIFICANT BYTE 151
MOST SIGNIFICANT BIT X POSITION
REGISTER 171
MSB 151
MÅNELANDING 132
NATURLIG LOGARITME 93
NEGATIVT TAL 49
NEW 27
NEXT 65
NIM 61
NOT 58
NUMERISK SÆT 81
NUMERISK Variabel 19
OKTAV 187
ON 90
OPEN 191
OPERATION 48

OPLØSNING I PRIMFAKTORER 87	RND 49
OR 57	ROD 48
ORDRE 14	ROM 111
OUT OF DATA 83	RUN 25
OVERFLOW 38	RUN STOP 28
OVERHØRING 42	RVS OFF 22
OVERSÆTTELSE 85	RVS ON 22
PAUSE 67	SAND 130
PEEK 116	SANDHEDSTABEL 139
POKE 116	SANDHEDSVÆRDI 130
POS 94	SAVE 98
POTENS 37	SAVTAK 184
PRIMFAKTOR 87	SCRATCH 194
PRIMTAL 68	SHIFT 19
PRINT 15	SIMULATION 92
PRINTER 97	SIN 92
PRIORITERING 36	SINUS 92
PROGRAM 24	SKIFTENØGLE 19
PROGRAMLINJE 24	SKÆRMKANT 117
PROGRAMLISTE 25	SKÆRMKODE 119
PROGRAMMERINGSSPROG 13	SLET 18
PSEUDOVARIABEL 91	SLETTETAST 18
PULSBREDDE 184	SMÅ BOGSTAVER 23
RAM 111	SPC 55
RAMTOP 151	SPRITE 168
RANDOM 49	SPRITENUMMER 169
RANDOM ACCESS MEMORY 111	SQR 48
READ 83	STØJ 184
READ ONLY MEMORY 111	STARTVÆRDI 67
READY 14	STEMME 187
REDIMENSIONED ARRAY 78	STEP 66
REDO FROM START 40	STOP 37
RELATION 52	STR\$ 73
RELEASE 183	STRENG 19
REM 95	STRENGDELING 71
REMARK 95	STRENGSÆT 78
RENAME 194	STRENGVARIABEL 19
RENTESREGNING 35	STRING TOO LONG 95
REPEAT 123	SUBROUTINE 76
RESERVERET ORD 18	SUSTAIN 183
RESTORE 83	SYNTAKS 17
RETURN 76	SYNTAKSFEJL 17
RIGHT\$ 69	SYNTAX ERROR 17

SYSTEMVARIABEL 117
SÆT 77
SØGEORD 82
TAB 55
TABEL 35
TABULATOR 55
TABULATORFUNKTION 55
TAN 93
TANGENS 93
TEKSTBEHANDLING 92
TEKSTFELT 117
TERNING 50
THEN 44
TI 50
TI\$ 50
TILFÆLDIGT TAL 49
TITALSSYSTEM 109
TOM STRENG 21
TONE 182
TONEVÆRDI 187
TOTALSSYSTEM 110
TREKANT 184
TRIGONOMETRI 92

TRIGONOMETRISK FUNKTION 92
TVÆRSUM 76
TYPE MISMATCH 40
TÆLLER 46
UENDELIG LØKKE 65
UNDEFINED FUNCTION 94
UNDEFINED STATEMENT 56
UNDERVISNINGSPROGRAM 196
UR 50
VAL 74
VARIABEL 16
VARIABELNAVN 17
VARIGHED 184
VERIFY 99
VICII 143
VIDEO INTERFACE CHIP 143
VÆRDI 16
VÆRDITILSKRIVNING 16
XMSB 171
ZX80 7

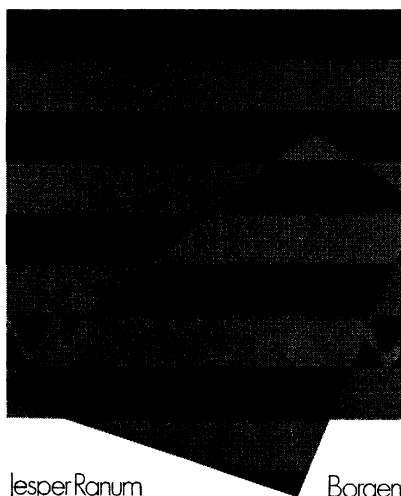
PS: Dette indeks har maskinen selv skrevet med et program fra bogen.

... Commodore 64 indeholder en komplet synthesizer.

Læs derfor også

SYNTHESIZERE

og andre elektroniske musikinstrumenter



Jesper Ranum
SYNTHESIZERE
og andre elektroniske
musikinstrumenter
280 sider illustreret

Jesper Ranum

Borgen

... Det her er bogen, jeg, og med garanti mange andre har ventet på i årevis. Bogen, der kan hjælpe musikeren til optimalt at forstå og udnytte synthesizeren ...

Opus nr. 1 1984

... et hovedværk (også internationalt set) ...

IBC lektørdtalelse

... bogen her er, som den første på dansk om emnet, en sikker vejviser – så hvad enten du ønsker bedre at forstå dit instrument, dine elevers eller børns musikoplevelser eller dine egne musikalske begrænsninger, så læs løs ...

Kenneth Knudsen

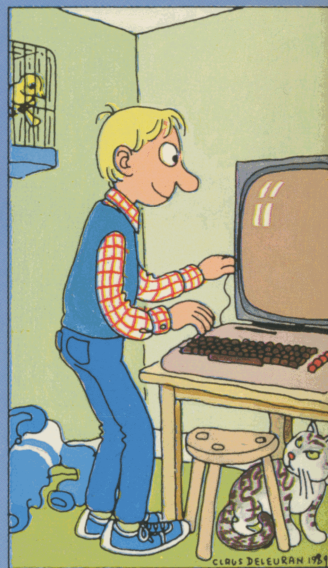
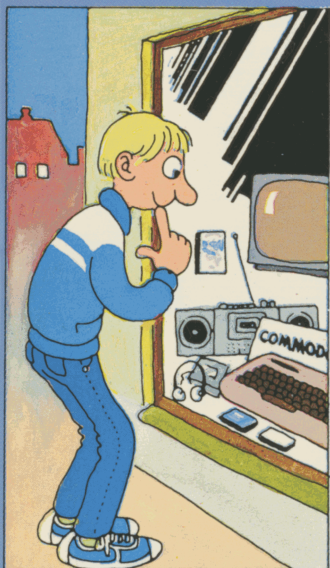
BORGEN

Med denne bog kan enhver lære at programmere . . .

Erwin Neutzsky-Wulffs nye bog, *BASIC med Commodore 64* er skrevet specielt til den fremragende og meget udbredte hjemmecomputer, Commodore 64, og indeholder et grundlæggende kursus i programmeringssproget BASIC.

Forfatteren starter helt fra grunden med maskinens betjening og fortæller detaljeret om de forskellige BASIC-ords betydning.

Med talrige eksempler vises, hvordan BASIC-programmerne bygges op, og hvordan man bedst udnytter maskinens avancerede muligheder. Det hele er skrevet i et letforståeligt sprog og på en sådan måde, at læseren gradvis opbygger et solidt grundlag for at forstå, hvad der foregår.



Bogen indeholder endvidere en række færdige programmer med spændende og underholdende spil og lege, som lige er klar til at blive tastet ind i maskinen.

Bogen er både velegnet som indføring for nybegyndere og som arbejds- og idebog for erfarne hobbyfolk.



**This was brought to you
from the archives of**

<http://retro-commodore.eu>